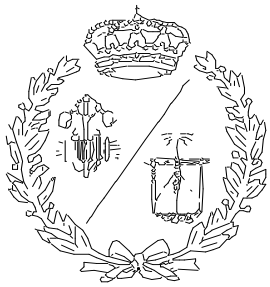
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

SISTEMA VIRTUAL DE ENTRENAMIENTO PARA LA DOCENCIA DE LAS ASIGNATURAS DE INGENIERÍA DE CONTROL

(Virtual Training System for Teaching Control Engineering Courses)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Ian Bejenaru

Tutor académico: Luciano Alonso Rentería

Julio - 2021

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

SISTEMA VIRTUAL DE ENTRENAMIENTO PARA LA DOCENCIA DE LAS ASIGNATURAS DE INGENIERÍA DE CONTROL

RESUMEN

En este trabajo se va a realizar un modelo de control para la regulación de la temperatura interior de una caja de madera. En el interior de la caja tendremos una resistencia calorífica que se encenderá o apagará mediante las instrucciones de un controlador PID que tratará de corregir el error del sistema en función de la temperatura objetivo que se quiere alcanzar y la temperatura real que se obtiene por realimentación a partir de un sensor de temperatura.

Utilizando Qt Designer para el diseño de la interfaz de usuario y mediante código Python se realizará una planta virtual de control de la temperatura. La interfaz gráfica permitirá tener una visión en tiempo real de la variación de temperatura. Incluirá un manual para que el alumno aprenda a manejar la interfaz y un libro de prácticas para que el alumno pueda asentar los conocimientos que ha visto en el aula.

VIRTUAL TRAINING SYSTEM FOR TEACHING CONTROL ENGINEERING COURSES

SUMMARY

In this work we are going to develop a control model for the regulation of the interior temperature of a wooden box. Inside the box we will have a heating resistor that will be turned on or off by the instructions of a PID controller that will try to correct the error of the system depending on the target temperature to be reached and the actual temperature obtained by feedback from a temperature sensor.

Using Qt Designer for the design of the user interface and using Python code, a virtual temperature control plant will be realized. The graphical interface will allow to have a real time view of the temperature variation. It will include a manual for the student to learn how to use the interface and a book of practices so that the student can consolidate the knowledge he has seen in the classroom.

1 INTRODUCCIÓN

1.1 ANTECEDENTES

Las plantas de laboratorio de la Universidad de Cantabria son una excelente herramienta para que los alumnos puedan practicar con ellas en asignaturas enfocadas en el control de procesos. Con la aparición del COVID-19 el acceso a los laboratorios ha sido restringido para evitar los contagios, se han tomado medidas como la de reducir el aforo o, en el peor de los casos, se ha prohibido la entrada a los laboratorios, por lo que los alumnos no han podido practicar con las plantas.

El mayor problema del COVID es el contagio entre personas, como las plantas del laboratorio son físicas, los alumnos siempre estarán expuestos a esta amenaza. Para que los alumnos no pierdan la oportunidad de practicar con las plantas del laboratorio, lo mejor será crear una planta virtual para evitar el contacto físico.

En este punto de la lectura pueden surgir preguntas del estilo ¿Se puede transformar una planta física a virtual? ¿Se puede reducir el riesgo al contagio entre personas? ¿Es posible hacer algo así?

1.1.1 Plantas de control de procesos

Para entender el funcionamiento del sistema en primer lugar, hay que saber lo que es una planta de control de procesos, puesto que el principal objetivo es tratar de replicar el funcionamiento de esta.

Las plantas de control de procesos son un conjunto de instalaciones hechas a medida para la realización de pruebas sobre sistemas de control de procesos, como pueden ser: detección y diagnóstico de fallo, planificación, optimización, modelización, etc. Todo esto se realiza en tiempo real, sobre sistemas simulados, a través de un sistema de monitorización y control guiado por un ordenador. Con las plantas de control se pueden simular procesos, de control de temperatura, control de caudal, control de entrada de gases, control de válvulas, etc.[1]

Figura 1.1. Planta de control de procesos industrial.[1]

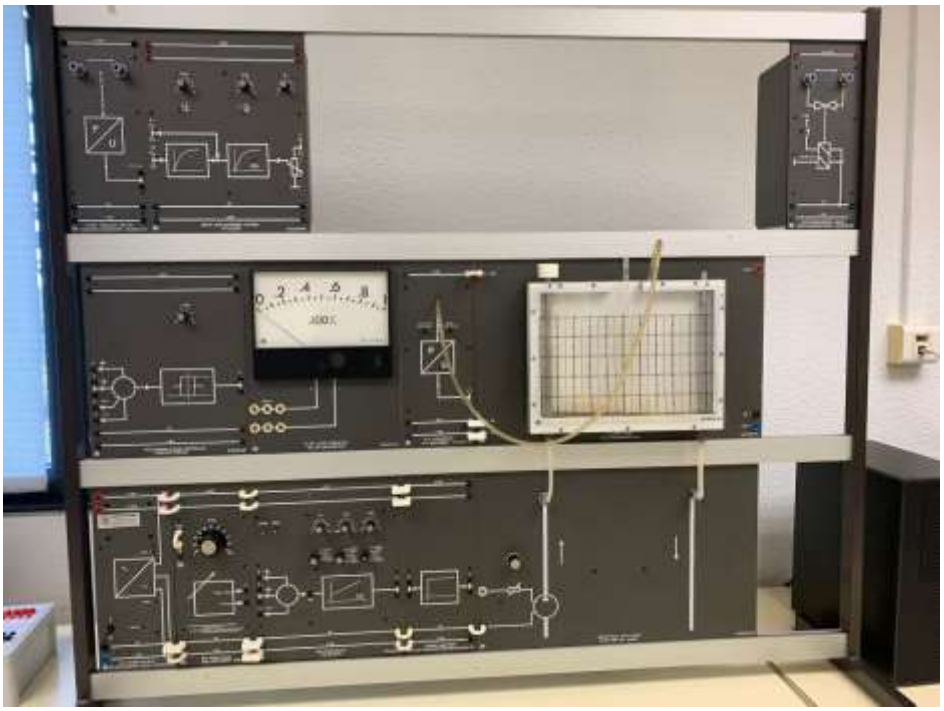


Figura 1.2. Planta de control de llenado de un tanque, Universidad de Cantabria.

Existen muchos tipos de plantas de control, pero la que más se asemeja al proyecto que se ha hecho es la planta de la **Figura 1.2.**

Una de las soluciones para reducir el riesgo de contagio entre personas es sustituir las plantas de control de procesos físicas por simuladores virtuales. Estos simuladores sirven como un recurso adicional para la docencia ya que los alumnos lo pueden utilizar en cualquier lugar y reforzar lo que han aprendido en las aulas.

1.1.2 Interfaz de usuario

Para resolver el objeto del proyecto se utiliza una interfaz gráfica de usuario con la que se simula el funcionamiento en tiempo real de una planta de control de temperatura. Para simplificar el término interfaz gráfica de usuario, se utilizará las siglas GUI (Graphic User Interface, GUI).

La interfaz gráfica de usuario es el espacio donde se producen las interacciones entre seres humanos y máquinas. Las interfaces básicas de usuario son aquellas que incluyen elementos con contenido gráfico que al interaccionar con ellas muestran unos resultados por pantalla. El principal objetivo de una interfaz de usuario es crear un contenido gráfico que sea agradable e intuitivo para que el usuario pueda interactuar con ello. [2]

Para el caso particular del proyecto, las principales funciones que desmpeñará la interfaz de usuario son las siguientes:

- Puesta en marcha y apagado del sistema de control.
- Control de los botones manipulables del equipo.
- Mostrar el diagrama de bloques del sistema.
- Información de la entrada y la salida del sistema en tiempo real.
- Simulación de posibles perturbaciones.
- Posibilidad de guardar datos para utilizarlos en otros programas.
- Simulación del calentamiento de los materiales que intenta imitar.

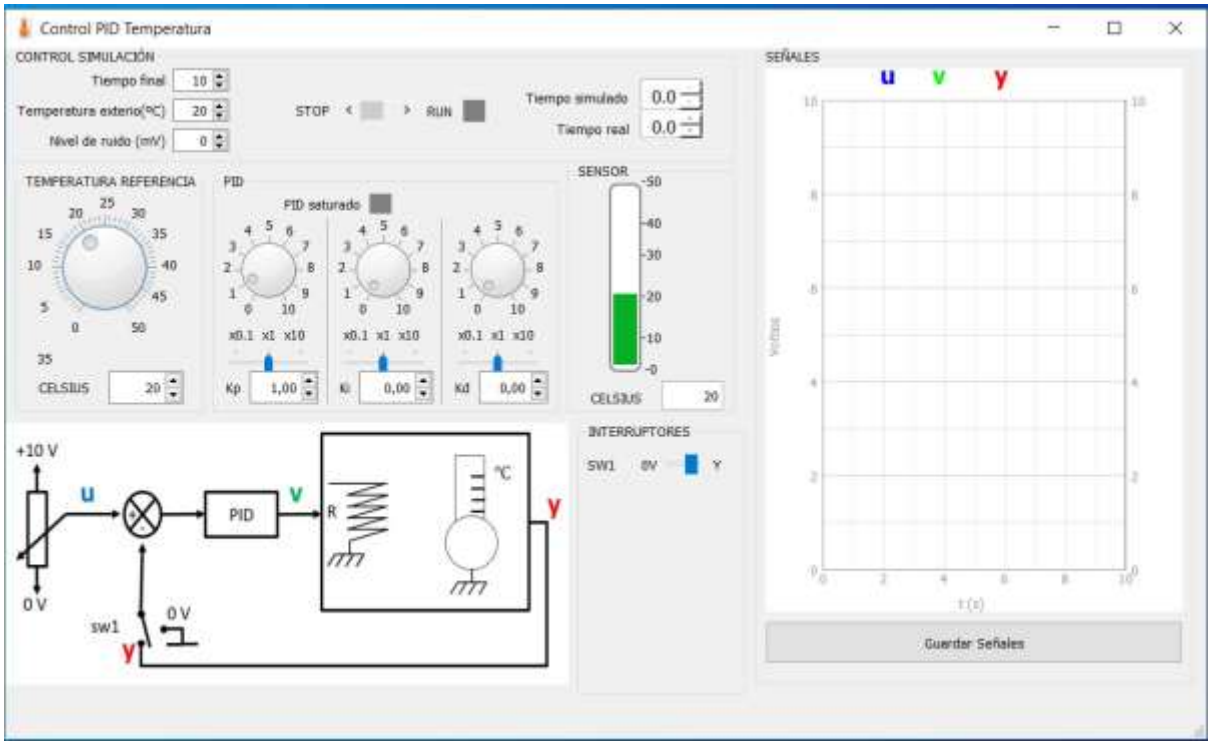


Figura 1.3. Interfaz gráfica de usuario del sistema de control.

1.1.3 Python Vs. Matlab

En este apartado se hará una comparativa entre dos lenguajes de programación muy utilizados en el ámbito de la ingeniería, Python y Matlab. Se verá lo mejor y lo peor de cada uno, se comparará las interfaces gráficas de usuario de cada uno y se decidirá cuál de los dos lenguajes es el óptimo para resolver el objeto del proyecto.

Popularidad de los lenguajes

La primera versión de Matlab fue lanzada en 1984, la de Python en 1991, siete años más tarde. Para ver la popularidad de los lenguajes de programación haremos una búsqueda en la página web "tiobe.com" en la que podemos observar que el 10 de junio de 2021 Python ocupa el segundo puesto de lenguaje de programación más popular en el mundo, según la comunidad de programación, mientras que Matlab ocupa el puesto diecinueve (ver **Figura 1.4**).[3]

Jun 2021	Jun 2020	Change	Programming Language	Ratings	Change
1	1		C	12.54%	-4.65%
2	3	↑	Python	11.84%	+3.48%
3	2	↓	Java	11.54%	-4.56%
4	4		C++	7.36%	+1.41%
5	5		C#	4.33%	-0.40%
6	6		Visual Basic	4.01%	-0.68%
7	7		JavaScript	2.33%	+0.06%
8	8		PHP	2.21%	-0.05%
9	14	⇈	Assembly language	2.05%	+1.09%
10	10		SQL	1.88%	+0.15%
11	19	⇈	Classic Visual Basic	1.72%	+1.07%
12	31	⇈	Groovy	1.29%	+0.87%
13	13		Ruby	1.23%	+0.25%
14	9	⇊	R	1.20%	-0.99%
15	16	↑	Perl	1.18%	+0.36%
16	11	⇊	Swift	1.10%	-0.35%
17	37	⇈	Fortran	1.07%	+0.80%
18	22	⇈	Delphi/Object Pascal	1.06%	+0.47%
19	15	⇊	MATLAB	1.05%	+0.15%

Figura 1.4. Índice TIOBE (10/06/2021).[3]

Similitud de los lenguajes

Ambos lenguajes de programación comparten múltiples ventajas como:

- Simplicidad a la hora de escribir código.
- Facilidad de comprensión a la hora de leer código.
- Multiplataforma (Windows, Linux, MacOS).

Diferencias entre los lenguajes

-Propósito: Python tiene un propósito general mientras que Matlab esta especialmente diseñado para ingenieros y científicos.

-Licencia: Python posee la “Python Software Foundation License” que cumple con los requisitos para ser declarada una licencia de software libre por lo que no hay que pagar una membresía para poder usar Python, mientras que Matlab posee una licencia “propietario” propia de cada usuario por lo que hay que pagar una membresía anual de 800 €, o si quieres una licencia permanente 2000 €. Python al ser un software libre permite acceso a su código fuente para que pueda ser estudiado, modificado y utilizado libremente. Matlab al ser un software de código cerrado no existe posibilidad de acceder a su código fuente por lo que, no permite su libre modificación ni tampoco se puede leer por parte de terceros.

-Documentación: Tanto Python como Matlab disponen de una página oficial donde se explica el funcionamiento de las diferentes instrucciones de código. La principal ventaja de que Python sea un software libre es permitir a los usuarios que puedan crear nuevas funciones y librerías para resolver algo más específico, pero esto es un arma de doble filo, porque al crear nuevas librerías, la documentación oficial de Python no explica el funcionamiento de esas nuevas instrucciones, dejando en manos del usuario dicha responsabilidad, esto puede provocar que nuevas instrucciones estén mal explicadas o que generen cierta confusión entre los usuarios. Con Matlab esto no ocurre porque es un software de código cerrado por lo que toda la documentación es oficial. Pero para compensar esa desventaja de Python hay que tener en cuenta que Python es más popular que Matlab por lo tanto dispone de muchos más foros en la web que Matlab y con estos foros los usuarios se ayudan entre sí formando una mayor comunidad permitiendo que exista una mayor fuente de documentación que con Matlab.

Comparativa entre líneas de código

Comparativa entre instrucciones básicas de Matlab vs. Python:[4]

- Uno de los principales inconvenientes de usar Python, es que hasta para realizar cálculos matemáticos básicos se requiere el uso de librerías. Como se puede apreciar en la **Figura 1.5.** Python utiliza más líneas de código para llegar al mismo resultado que Matlab por el mero hecho de tener que llamar a una librería.

Python

```
>>> import numpy as np

# Create row vector
>>> row = np.array([1, 2, 3])
>>> row
array([1, 2, 3])

# Transpose
>>> col = row.T

# Compute inner product
>>> inner = np.dot(row,col)
>>> inner
14

# Compute outer product
>>> outer = np.dot(col,row)
>>> outer
14
```

MATLAB

```
% Create row vector
>> row = [1 2 3]
row =
     1     2     3

% Transpose
>> col = row';

% Compute inner product
>> inner = row*col
inner =
    14

% Compute outer product
>> outer = col*row
outer =
     1     2     3
     2     4     6
     3     6     9
```

Figura 1.5. Operaciones matemáticas de matrices, Python vs. Matlab.[5]

- Para otro tipo de instrucciones el código de Python puede ser menos detallado que el código de Matlab, ocupando menos espacio y haciendo más simple su interpretación, podemos apreciar esta sencillez en la **Figura 1.6.** Hay que tener en cuenta que para el ejemplo de la **Figura 1.6.** el código de Matlab es aproximadamente el doble de largo que el de Python, al ser el código más largo existe la posibilidad de cometer más errores tipográficos y se tarda más tiempo en leer e interpretar el código. En términos generales el código de Python es casi siempre más pequeño (refiriéndonos a líneas de código) que el código de Matlab que realiza la misma tarea.

Matlab:

```
count(max(floor(x),1),max(floor(y),1))= ...
count(max(floor(x),1),max(floor(y),1)) + 1
```

Python:

```
count[max(floor(x),0),max(floor(y),0)]+= 1
```

Figura 1.6. Simplicidad de programación, Matlab vs. Python.[6]

Python:

```
In[1]: n=3
In[2]: [1,2,4,7,11,14,16,17][n]
Out[2]: 7
```

Matlab:

```
>> n= 4 % Matlab indices start from 1
>> [1,2,4,7,11,14,16,17](n)
Error: Unbalanced or unexpected parenthesis or bracket
```

Figura 1.7. Tolerancia a la hora de usar instrucciones de código, Python vs. Matlab.[6]

En líneas generales se puede sacar la siguiente conclusión al comparar estos dos programas: Python tiene mejor legibilidad por lo que conduce a menos errores (ver **Figura 1.6**), tiene una depuración más rápida cuando se introducen errores porque es más tolerable a la hora de escribir código (ver **Figura 1.7** y una comprensión más rápida cuando uno debe trabajar con el código de otra persona (ver **Figura 1.6**).

Interfaz gráfica de usuario (GUI)

Se hará una pequeña comparativa entre la interfaz gráfica de usuario, comúnmente conocidas como GUI cuyas siglas provienen del inglés “Graphical User Interface”, que podemos crear con Python y Matlab.

Para crear una GUI en Matlab se necesita usar la instrucción “guide”, en la ventana de comandos. Se abrirá la ventana de creación y diseño de GUI (ver **Figura 1.8**), “guide” se solía utilizar en versiones antiguas de Matlab, por lo que se ha quedado obsoleto, en la actualidad se utiliza la herramienta de Matlab “App Designer” (ver **Figura 1.9**) que tiene nuevos componentes, más modernos y avanzados que la herramienta “guide”.Ocurre lo mismo para crear una GUI en Python solo que en este caso Python utiliza un programa llamado Qt Designer (ver **Figura 1.10**) con el que se puede crear y diseñar una GUI. Las GUI que se crean en el programa Qt Designer se relacionan con Python mediante la librería “pyqt5”.

Si se compara la herramienta “guide” de Matlab con el programa Qt Designer, se puede apreciar que “guide” es bastante básica; pero si se compara la herramienta “App Designer” con el programa Qt Designer se observa que son bastante similares en cuanto a los componentes que se puede utilizar y que incluso “App Designer” es algo más completo que Qt Designer. En definitiva, tanto con la aplicación de diseño de Matlab como con la de Python se puede crear una interfaz de usuario que sirva.

La principal ventaja de usar aplicaciones enfocadas en la creación y diseño de GUI es que se puede crear dichas GUI de forma rápida y sencilla, ahorrando el escribir líneas de código. El lenguaje gráfico que se utiliza en estas aplicaciones es muy similar porque nos ofrecen componentes gráficos equivalentes con nombres muy similares (menús, botones, etiquetas, listas seleccionables, etc.)

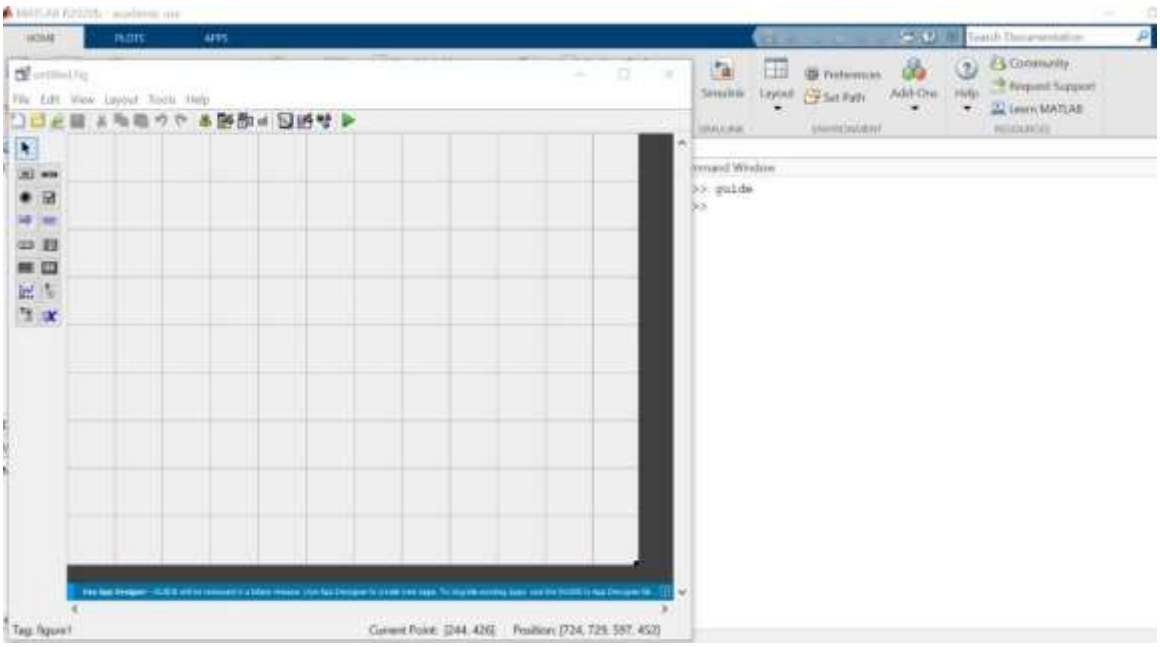


Figura 1.8. Guide, en Matlab.

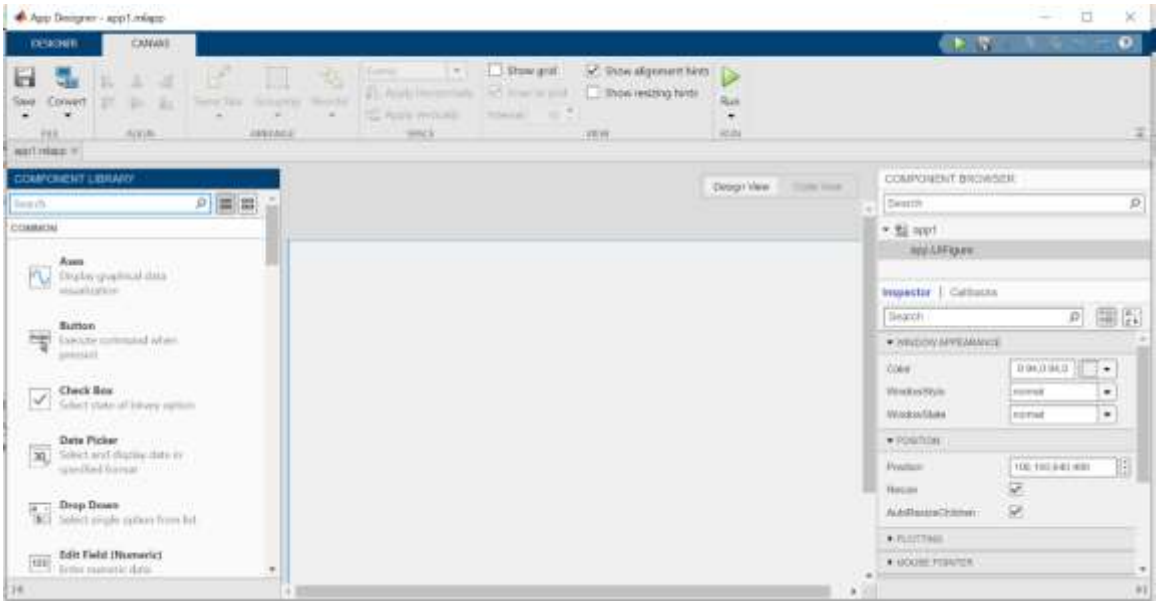


Figura 1.9. App Designer, en Matlab.

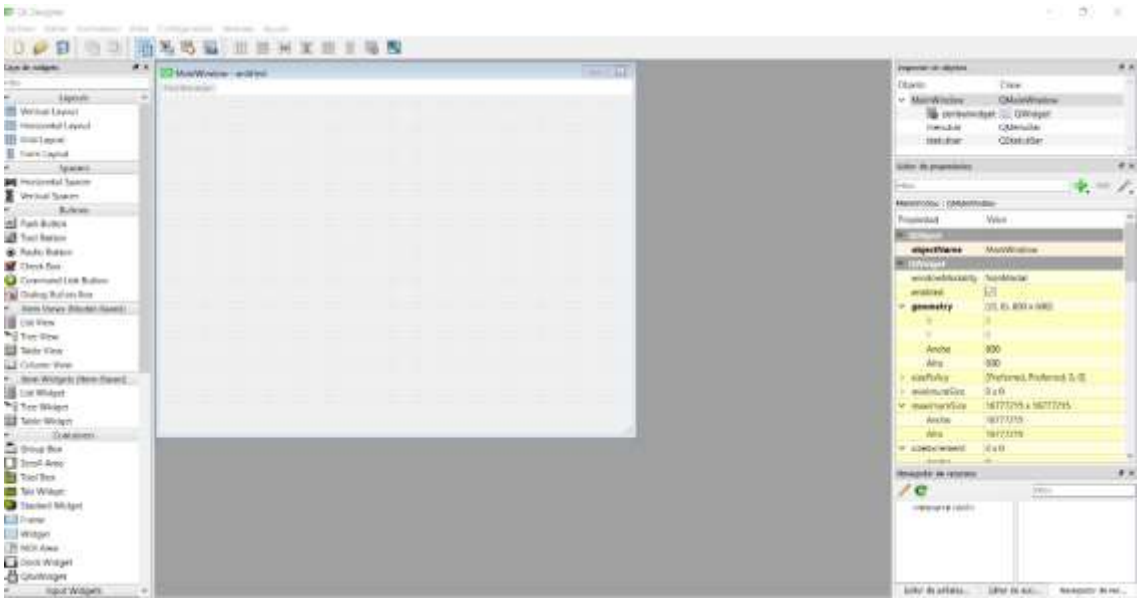


Figura 1.10. Herramienta de diseño y creación de interfaz de usuario (GUI) Qt Designer para Python.

Requisitos para la interfaz gráfica de usuario

Una vez visto las principales diferencias, y similitudes, entre Python y Matlab, toca enfocarse en analizar ¿Cuál será el mejor programa para desarrollar el objetivo del proyecto? Para ello se tiene que responder a una pregunta previa ¿Qué características principales se necesita para que la GUI sea lo más parecido a una planta de control de procesos real? Respuesta: que la simulación del proceso permita recopilar datos en tiempo real. Tanto Matlab como Python permiten recopilar datos, pero … ¿esos datos que recopilan son en tiempo real?

En Matlab para poder construir y simular modelos de sistemas físicos y sistemas de control mediante diagramas de bloque se debe utilizar Simulink (plataforma de simulación). Tanto con Python como con Simulink se puede realizar una simulación instantánea pero también se puede realizar una simulación en tiempo real.

En estos momentos surge una incógnita ¿Por qué es un inconveniente que la simulación no se haga en tiempo real?:

- No replicaría el funcionamiento de una planta de control de procesos real.
- Si la simulación es instantánea no se podría apreciar que le sucede al sistema si se le añade algún tipo de perturbación durante la simulación.

En vista de todas las comparativas parece que se puede hacer la misma GUI tanto con Matlab como con Python, pero Matlab tiene un problema a la hora de realizar operaciones en tiempo real, como la GUI tiene muchos botones interactivos, que estarán activos durante la simulación, si varían constantemente, Matlab no será capaz de procesar todos los movimientos que se hacen y si fuera capaz de procesar todos los movimientos, no sería en tiempo real porque tardaría en procesar la información. Esto hace que Matlab se vuelva un programa ineficiente para el proyecto.

¿Por qué Python?

El ganador es Python porque dispone de una serie de ventajas que hay que tener en cuenta:

- Es de software libre por lo que no hay que pagar una membresía para poder utilizarlo, muy bien documentado y con una gran comunidad de desarrolladores. No precisa de una licencia privativa y puede utilizarse sin problemas en los lugares que no existe conexión a internet.
- Cuenta con múltiples paquetes para el desarrollo de objetos gráficos e interfaces de usuario.
- Dispone de la colección de software de computación científica SciPy que permite trabajar con matrices n-dimensionales mediante el paquete Numpy [7] así como producir figuras de alta calidad haciendo uso de la librería Matplotlib [8].
- La interfaz gráfica de Qt es más rápida que la de Matlab porque las librerías de Qt están desarrolladas en C++, que es un lenguaje compilado, mientras que las de Matlab están hechas en Java, que es un lenguaje interpretado.

En vista de todas las comparativas hechas entre Python y Matlab, podemos decantarnos por un vencedor, Python. La principal característica por la que se escoge Python es por su simulación en tiempo real, ya que en Matlab se presentan muchos retardos al simular la GUI.

1.1.4 Herramientas de desarrollo

En este apartado se verá las herramientas que se necesitan para desarrollar el proyecto, Qt Designer y Python.

Qt Designer

Desarrollado por dos ingenieros de software noruegos Haavard Nord y Eirik Chambe-Eng, el marco Qt se puso a disposición del público por primera vez en mayo de 1995. La primera versión pública de Qt fue lanzada por una empresa llamada Trolltech.[9]

Qt Designer es la herramienta de Qt para diseñar y construir interfaces gráficas de usuario (GUI) con Qt Widgets. Los widgets y formularios creados con Qt Designer se integran perfectamente con el código programado, utilizando el mecanismo de señales y ranuras de Qt, de modo que se puede asignar fácilmente el comportamiento a los elementos gráficos. Todas las propiedades establecidas en Qt Designer pueden modificarse dinámicamente dentro del código. Además, características como la promoción de widgets y los plugins personalizados le permiten utilizar sus propios componentes con Qt Designer.[10]

Figura 1.11. Logotipo de Qt Designer.

Se utilizará Qt Designer para crear la GUI ya que es una herramienta intuitiva que se utiliza con facilidad y combina muy bien con el código programado.

Python

Para poder crear una GUI se utilizará Qt Designer y para programar la funcionalidad de la interfaz se utilizará el lenguaje de programación de Python. Python podrá interactuar con la GUI, creada en Qt Designer, mediante el uso de la librería “pyqt5”, esta librería conecta Qt Designer con Python.

Python es un lenguaje de programación interpretado de código abierto, orientado a objetos. Tiene una sintaxis sencilla y cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único. Python fue creado a finales de los ochenta por Guido van Rossum. [11]

Figura 1.12. Logotipo de Python.[12]

1.2 OBJETIVOS

El objetivo principal de trabajo es el desarrollo de un software de simulación de un sistema de control PID para la regulación de la temperatura. Se empleará en la docencia de las asignaturas del área de ingeniería de control. Las características principales son las siguientes:

- Funcionamiento en tiempo real, a diferencia de otros programas de simulación como Matlab/Simulink.
- Interfaz gráfica con componentes visuales próximos a los que se encontrarían en un sistema real, lo que permite una mejor comprensión de la arquitectura física de un sistema de control real, y su relación con la representación matemática del mismo (función de transferencia).
- Controles interactivos de los parámetros de la simulación y el control, lo que permite variar estos parámetros en tiempo real, así como visualizar de forma realista los efectos de esas variaciones sobre las variables de interés del proceso.
- Visualización gráfica en tiempo real de las señales de interés, con posibilidad de exportar dichas señales para su posterior análisis con Matlab.

Este software se acompaña de un manual de usuario, así como de un libro de prácticas, para que el alumno pueda asentar los conocimientos adquiridos en el aula.

1.3 ESTRUCTURA DEL DOCUMENTO

El primer capítulo de este trabajo es esta introducción donde se ha descrito con detalle la motivación que ha llevado a considerar necesario realizar el desarrollo de un sistema virtual de entrenamiento para la docencia de las asignaturas de control de procesos, se han presentado las herramientas que serán necesarias para el desarrollo del proyecto. Además, se han planteado los objetivos detallados de este trabajo y se incluye este resumen.

En el segundo capítulo se verá la estructura general que tiene un sistema de control realimentado ya que este tipo de sistema de control es el que se utilizará para resolver el objeto del proyecto. Se verá el sistema propuesto con las funciones que realiza y se escogerá cada uno de los componentes para llevar a cabo el objeto del proyecto.

El tercer capítulo se discretizan las ecuaciones que simulan el funcionamiento del proceso del sistema, se preparan las ecuaciones para que se puedan implementar en el código Python. Se habla sobre la

interfaz gráfica de usuario, sus principales características y funcionamiento. En último lugar se menciona la estructura que sigue el código Python para simular el funcionamiento de la planta térmica y sobre cómo se debe utilizar el ejecutable de la aplicación para que se pueda utilizar en cualquier dispositivo.

En el cuarto capítulo se verá cuáles son los trabajos futuros que se pueden hacer con el proyecto actual, mejoras y variantes que se pueden implementar.

El quinto capítulo se mencionan las conclusiones y observaciones que se han obtenido con la realización de este proyecto.

En el capítulo siete se encuentran las referencias bibliográficas que se han utilizado para complementar la información de algunos capítulos.

El capítulo número ocho muestra las líneas de código por las que están formadas la aplicación que simula el funcionamiento de una planta térmica.

2 SISTEMAS DE CONTROL REALIMENTADO

En este apartado se verá la estructura general que tiene un sistema de control realimentado ya que éste tipo de sistema de control es el que se utilizará para resolver el objeto del proyecto. Se verá el sistema propuesto con las funciones que realiza y se escogerá cada uno de los componentes para llevar a cabo el objeto del proyecto.

Es conveniente dar una definición a lo que es un sistema de control. Para responder, se puede decir que en nuestra vida diaria existen numerosos objetivos que necesitan cumplirse. Por ejemplo, en el ámbito doméstico, se requiere regular la temperatura y humedad de las casas y edificios para tener un ambiente cómodo. Para transportación, se requiere controlar que un automóvil o un aeroplano se muevan de un lugar a otro en una forma segura y exacta. En la industria, los procesos de manufactura tienen un sinnúmero de objetivos para productos que satisfacen requerimientos de precisión y costo. La búsqueda para alcanzar tales "objetivos" requiere normalmente utilizar un sistema de control que implante ciertas estrategias de control. En años recientes, los sistemas de control han asumido un papel cada vez más importante en el desarrollo y avance de la civilización moderna y la tecnología. Prácticamente, cada aspecto de las actividades de nuestra vida diaria está afectado por algún tipo de sistema de control. Los sistemas de control se encuentran en gran cantidad en todos los sectores de la industria, tales como control de calidad de los productos manufacturados, líneas de ensamble automático, control de máquinas-herramienta, tecnología espacial y sistemas de armas, control por computadora, sistemas de transporte, sistemas de potencia, robótica y muchos otros. Incluso el control de inventarios y los sistemas económicos y sociales se pueden visualizar a través de la teoría de control automático.[13]

En general, el objetivo de un sistema de control es controlar las salidas en alguna forma prescrita mediante las entradas a través de los elementos del sistema de control.

Se pueden clasificar los sistemas de control en dos tipos:

- Sistemas de control de lazo abierto: la salida no interviene en la acción de control.
- Sistemas de control de lazo cerrado: es necesario conocer el valor de la salida porque interviene en la acción de control.

El motivo de utilizar realimentación es para reducir el error entre la entrada de referencia y la salida del sistema. La reducción del error del sistema es sólo uno de los efectos más importantes que la realimentación realiza sobre el sistema, porque también tiene efectos en características del

desempeño del sistema como la estabilidad, ancho de banda, ganancia global, perturbaciones y sensibilidad.[13]

2.1 ESTRUCTURA GENERAL

En los sistemas de retroalimentación, la variable que se controla (como la temperatura o la velocidad) se mide con un sensor y la información medida se devuelve al controlador para influir en la variable controlada. El principio se ilustra fácilmente con un sistema muy común, la caldera doméstica controlada por un termostato (ver **Figura 2.1**). Esta imagen identifica las partes principales del sistema y muestra las direcciones del flujo de información de un componente a otro.

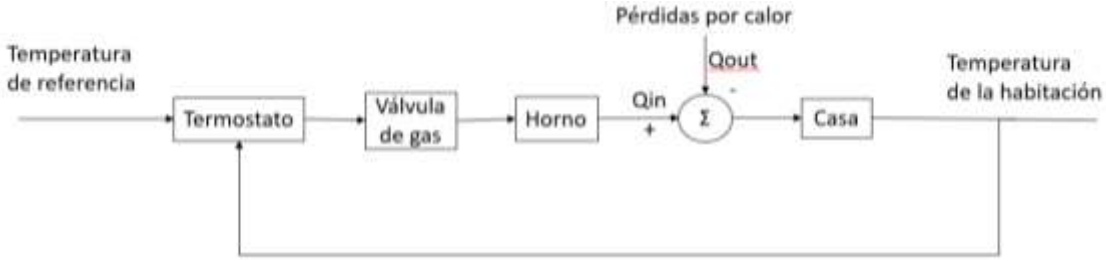


Figura 2.1. Diagrama de bloques de los componentes de un sistema de control de la temperatura ambiente.

Supongamos que tanto la temperatura de la habitación en la que se encuentra el termostato como la temperatura exterior son significativamente inferiores a la temperatura de referencia (también llamada punto de ajuste) cuando se aplica la energía. El termostato estará encendido y la lógica de control abrirá la válvula de gas del horno y encenderá la caja de fuego. Esto hará que el calor Qin se suministre a la casa a un ritmo que será significativamente mayor que la pérdida de calor Qout. Como resultado, la temperatura de la habitación aumentará hasta que supere el ajuste de referencia del termostato en una pequeña cantidad. En ese momento, la caldera se apagará y la temperatura ambiente comenzará a descender hacia el valor exterior. Cuando caiga una pequeña cantidad por debajo del valor de referencia, el termostato se encenderá de nuevo y el ciclo se repetirá. A partir de este ejemplo, podemos identificar los componentes genéricos del sistema elemental de control por retroalimentación (ver **Figura 2.2**) [14]

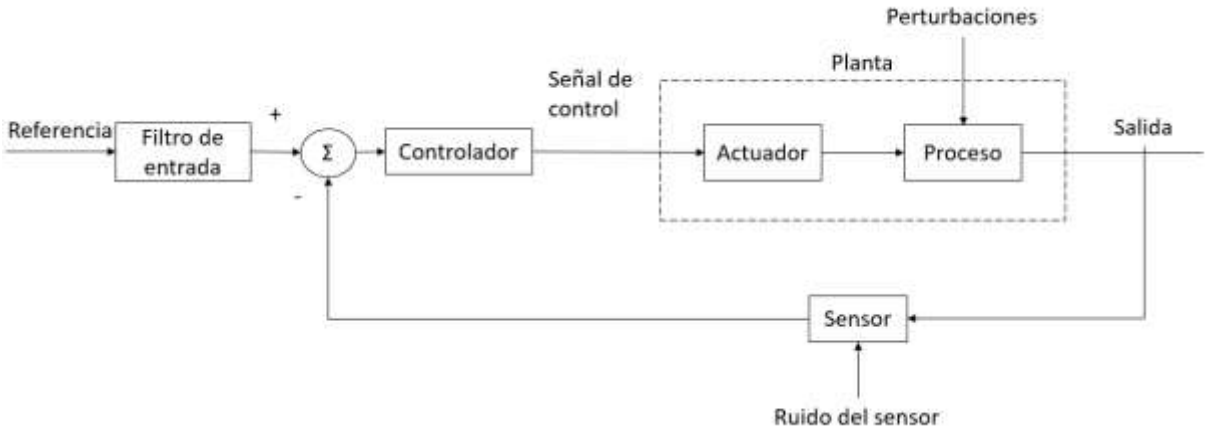


Figura 2.2. Diagrama de bloques de los componentes de un control de retroalimentación elemental.

El componente central de un sistema de control con retroalimentación es el proceso cuya salida debe ser controlada. En el ejemplo anterior (ver **Figura 2.1**), el proceso sería la casa cuya salida es la temperatura de la habitación y la perturbación del proceso es el flujo de calor de la casa debido a la conducción a través de las paredes y el techo hacia la temperatura exterior más baja. (El flujo de calor hacia el exterior también depende de otros factores como el viento, las puertas abiertas, etc.) El actuador es el dispositivo que puede influir en la variable controlada del proceso y, en nuestro caso, el actuador es un horno de gas. En realidad, el horno suele tener una luz piloto o un mecanismo de encendido, una válvula de gas y un ventilador que se enciende o apaga en función de la temperatura del aire en el horno. Estos detalles ilustran el hecho de que muchos sistemas de retroalimentación contienen componentes que a su vez forman otros sistemas de retroalimentación. El problema central del actuador es su capacidad para mover la salida del proceso con la velocidad y el alcance adecuados. El horno debe producir más calor del que pierde la casa en el peor día y debe distribuirlo rápidamente si se quiere mantener la temperatura de la casa en un rango estrecho. La potencia, la velocidad y la fiabilidad suelen ser más importantes que la precisión. Por lo general, el proceso y el actuador están íntimamente conectados y el diseño de control se centra en encontrar una entrada o señal de control adecuada para enviar al actuador. La combinación de proceso y actuador se denomina planta y el componente que realmente calcula la señal de control deseada es el controlador. Debido a la flexibilidad del procesamiento de señales eléctricas, el controlador suele trabajar con señales eléctricas, aunque el uso de controladores neumáticos basados en aire comprimido ocupa un lugar importante desde hace mucho tiempo en el control de procesos. El componente etiquetado como termostato en la **Figura 2.1** mide la temperatura ambiente y se denomina sensor en la **Figura 2.2**, un dispositivo cuya salida contiene inevitablemente ruido de sensor. La selección y la ubicación de los

sensores son muy importantes en el diseño del control, ya que a veces no es posible que la verdadera variable controlada y la variable detectada sean la misma. Por ejemplo, aunque realmente queramos controlar la temperatura de la casa en su conjunto, el termostato está en una habitación concreta, que puede estar o no a la misma temperatura que el resto de la casa. Por ejemplo, si el termostato está ajustado a 68 °F, pero está colocado en el salón, cerca de una chimenea, una persona que trabaje en el estudio podría seguir sintiendo un frío incómodo. Además de la colocación, las propiedades importantes de un sensor son la precisión de las mediciones, así como el bajo ruido, la fiabilidad y la linealidad. El sensor suele convertir la variable física en una señal eléctrica para que la utilice el controlador. Nuestro sistema general también incluye un filtro de entrada cuya función es convertir la señal de referencia en forma eléctrica para su posterior manipulación por el controlador. En algunos casos, el filtro de entrada puede modificar la entrada del comando de referencia de forma que mejore la respuesta del sistema. Por último, hay un comparador que calcula la diferencia entre la señal de referencia y la salida del sensor para dar al controlador una medida del error del sistema.[14]

Un sistema de control con realimentación básico tiene los siguientes elementos: referencia, filtro de entrada, comparador, controlador, actuador, proceso, planta, perturbaciones, salida y un sensor.

- Referencia: es la entrada que se le da al sistema y responde a esta pregunta: ¿Qué es lo que se quiere conseguir?
- Filtro de entrada: convierte la variable de entrada a una variable eléctrica para que pueda ser interpretada por el controlador.
- Comparador: obtiene el error del sistema, es la diferencia entre la referencia y la salida.
- Controlador: se encarga de leer los datos que le llegan y de elaborar una respuesta para el actuador.
- Actuador: es el dispositivo que puede influir en la variable controlada del proceso.
- Proceso: es el componente central de un sistema de control con retroalimentación y cuya salida debe ser controlada
- Planta: combinación de proceso y actuador.
- Perturbaciones: variables externas del sistema que pueden afectar a la hora de elaborar una respuesta de control.

- Salida: es la respuesta del sistema, el objetivo es que sea lo más parecida a la señal de referencia.
- Sensor: convierte la variable física en una señal eléctrica para que la utilice el controlador. Es un dispositivo que inevitablemente contiene ruido.

2.1.1 Representaciones matemáticas de los sistemas de control

Los estudios de los sistemas de control dependen fuertemente del uso y aplicación de las matemáticas. Requerimientos matemáticos tales como: la teoría de la variable compleja, ecuaciones diferenciales, transformada de Laplace, transformada Z y función de transferencia.

Variable compleja

Un número complejo tiene una parte real y una parte imaginaria, ambas son constantes. Si la parte real o la parte imaginaria son variables, el número complejo se denomina variable compleja. En la transformada de Laplace, usamos la notación s como una variable compleja. Una variable compleja s tiene dos componentes: una componente real σ y una imaginaria ω. En forma gráfica, la componente real de s está representada por el eje σ en la dirección horizontal y la componente imaginaria se mide a lo largo del eje vertical $j\omega$, en el plano complejo s. La **Figura 2.3** ilustra el plano complejo s, en donde cualquier punto arbitrario $s = s_1$ está definido por las coordenadas $\sigma = \sigma_1$, y $\omega = \omega_1$, o simplemente $s_1 = \sigma_1 + j\omega_1$.[13]

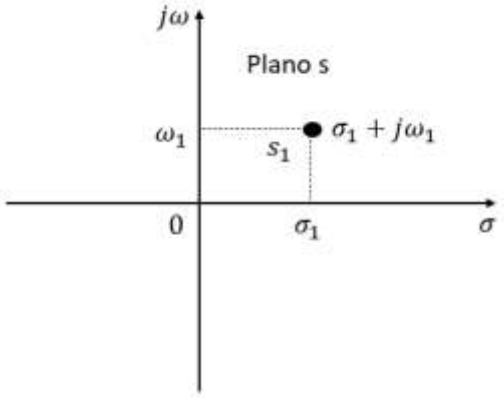


Figura 2.3. Plano complejo s.

Se dice que la función $G(s)$ es una función de la variable compleja s, si para cada valor de s existen uno o más valores correspondientes de $G(s)$. Debido a que s se define con partes real e imaginaria, la función $G(s)$ también esté representada por sus partes real e imaginaria:

$$G(s) = Re\ G(s) + j\ Im\ G(s) \quad [2.1]$$

Donde $Re\ G(s)$ denota la parte real de $G(s)$ e $Im\ G(s)$ representa la parte imaginaria de $G(s)$. La función $G(s)$ también está representada mediante el plano complejo $G(s)$, con $Re\ G(s)$ como el eje real e $Im\ G(s)$ como el imaginario. Si para cada valor de s existe sólo un valor correspondiente de $G(s)$ en el plano $G(s)$, se dice que $G(s)$ es una función univaluada, y el mapeo de los puntos en el plano s dentro de los puntos del plano $G(s)$ se describe como un solo valor. Si el mapeo desde el plano $G(s)$ al plano s también es un valor sencillo, el mapeo se denomina uno a uno. Sin embargo, existen muchas funciones cuyo mapeo desde el plano de función al plano de la variable compleja no es un solo valor. Por ejemplo, dada la siguiente función:

$$G(s) = \frac{1}{s(s+1)} \quad [2.2]$$

Es aparente que para cada valor de s, existe sólo un valor correspondiente para $G(s)$. Sin embargo, para el mapeo inverso esto no es verdad, por ejemplo, el punto $G(s) = \infty$ está mapeado en dos puntos, $s = 0$ y $s = -1$, en el plano s.[13]

Ecuaciones diferenciales

Una gran variedad de sistemas en ingeniería se modelan matemáticamente mediante ecuaciones diferenciales. Estas ecuaciones generalmente involucran derivadas e integrales de variables dependientes con respecto a la variable independiente. Por ejemplo, un circuito eléctrico *RLC* en serie (resistencia-inductancia-capacitancia) se puede representar por la ecuación diferencial:

$$Ri(t) + L\frac{di(t)}{dt} + \frac{1}{C}\int i(t)dt = e(t) \quad [2.3]$$

En donde R es la resistencia, L la inductancia, C la capacitancia, $i(t)$ la corriente en la red y $e(t)$ el voltaje aplicado. En este caso, $e(t)$ es la función de excitación, t la variable independiente e $i(t)$ la variable dependiente o desconocida que será determinada al resolver la ecuación diferencial. La ecuación 2.3 se denomina ecuación diferencial de segundo orden, y hace referencia al sistema como un sistema de segundo orden. Muchos sistemas físicos son no lineales y se deben describir mediante ecuaciones diferenciales no lineales. Por ejemplo, la ecuación diferencial que describe el movimiento de un péndulo (ver **Figura 2.4**) es:

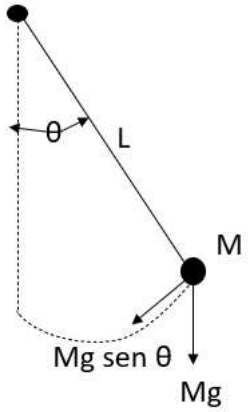


Figura 2.4. Péndulo simple.

$$ML\frac{d^2\theta(t)}{dt^2} + Mg \cdot sen\,\theta(t) = 0 \qquad [2.4]$$

Ya que $\theta(t)$ aparece con una función seno, la ecuación 2.4 es no lineal, y el sistema se conoce como sistema no lineal.[13]

Ecuaciones de estado

En general, una ecuación de n-ésimo orden se puede descomponer en n ecuaciones de primer orden. Ya que, en principio, las ecuaciones diferenciales de primer orden son más fáciles de resolver que otras de orden más alto. Para la ecuación 2.3, se tiene:

$$x_1(t) = \int i(t)dt \qquad [2.5]$$

$$x_2(t) = \frac{dx_1(t)}{dt} = i(t) \qquad [2.5]$$

La ecuacion 2.3 se descompone en las siguientes dos ecuaciones diferenciales de primer orden:

$$\frac{dx_1(t)}{dt} = x_2(t) \qquad [2.6]$$

$$\frac{dx_2(t)}{dt} = -\frac{1}{LC}x_1(t) - \frac{R}{L}x_2(t) + \frac{1}{L}e(t) \qquad [2.7]$$

En la teoría de los sistemas de control, el conjunto de ecuaciones diferenciales de primer orden de la ecuacion a analizar se conoce como ecuaciones de estado, y $x_1, x_2, \ldots, x_n$, son llamadas variables de estado. El estado de un sistema se refiere a las condiciones pasadas, presentes y futuras del sistema. Desde un sentido matemático, es conveniente definir un conjunto de variables de estado y ecuaciones de estado para modelar sistemas dinámicos.[13]

Transformada de Laplace

La transformada de Laplace es una de las herramientas matemáticas utilizadas para la solucion de ecuaciones diferenciales ordinarias lineales. La transformada de Laplace convierte la ecuacion diferencial en una ecuacion algebraica en s. Entonces es posible manipular la ecuacion algebraica mediante reglas algebraicas simples, para obtener la solución en el dominio s. La solución final se obtiene tomando la transformada inversa de Laplace.[13]

Mediante el uso de la transformada de Laplace, es posible convertir muchas funciones comunes, tales como las funciones senoidales, las funciones senoidales amortiguadas y las funciones exponenciales, en funciones algebraicas de una variable s compleja. Las operaciones tales como la diferenciación y la integración se sustituyen mediante operaciones algebraicas en el plano complejo. Por tanto, en una ecuación algebraica, una ecuación diferencial lineal se transforma en una variable compleja s. Si se resuelve la ecuación algebraica en s para la variable dependiente, la solución de la ecuación diferencial (la transformada inversa de Laplace de la variable dependiente) se encuentra mediante una tabla de transformadas de Laplace o una técnica de expansión en fracciones parciales. Una ventaja del método de la transformada de Laplace es que permite el uso de técnicas gráficas para predecir el desempeño del sistema, sin tener que resolver las ecuaciones diferenciales del sistema. Otra ventaja del método de la transformada de Laplace es que, cuando se resuelve la ecuación diferencial, es posible obtener simultáneamente tanto el componente transitorio como el componente de estado estable de la solución. [15]

La transformada de Laplace de f(t) se obtiene mediante:

$$\mathcal{L}[f(t)\} = F(s) = \int_0^\infty e^{-st}\, dt[f(t)] = \int_0^\infty f(t)e^{-st}\, dt \qquad [2.8]$$

El proceso inverso de encontrar la función del tiempo f(t) a partir de Laplace F(s) se denomina transformada inversa de Laplace. La notacion para la transformada inversa de Laplace es $\mathcal{L}^{-1}$, se encuentra a partir de F(s) mediante la siguiente integral de inversión:

$$\mathcal{L}^{-1}[F(s)\} = f(t) = \frac{1}{2\pi j}\int_{c-j\infty}^{c+j\infty} F(s)e^{st}\, ds, \qquad para\ t > 0 \qquad [2.9]$$

Transformada Z

La transformada z representa un método operacional para resolver ecuaciones en diferencias y sistemas lineales con datos discretos o digitales.[13]

Función de transferencia

La forma clásica de modelar sistemas lineales es utilizar funciones de transferencia para representar las relaciones entrada-salida entre variables.

La función de transferencia de un sistema descrito mediante una ecuación diferencial lineal e invariante con el tiempo se define como el cociente entre la transformada de Laplace de la salida Y(s) (función de respuesta) y la transformada de Laplace de la entrada U(s) (función de excitación) bajo la suposición de que todas las condiciones iniciales son cero. La función de transferencia G(s) se relaciona con la transformada de Laplace de la entrada y la salida a través de la siguiente relación:

$$G(s) = \frac{Y(s)}{U(s)} \qquad [2.10]$$

Con todas las condiciones iniciales puestas a cero, Y(s) y U(s) son las transformadas de Laplace de y(y) y u(t), respectivamente.[15]

2.2 SISTEMA PROPUESTO

En este apartado se verá la construcción del problema, lo que se quiere hacer y lo que se quiere controlar. Se resolverán preguntas del estilo: ¿Qué componentes se necesitan? ¿Cómo se controla el sistema? ¿Cuál es la entrada y la salida? Se dejará todo preparado para poder hablar del software que se quiere implementar con Python.

2.2.1 Funcionamiento del sistema

Para entender el funcionamiento del sistema en primer lugar se debe crear un diagrama de bloques que represente cada uno de los componentes que se utilizarán (ver **Figura 2.5**).

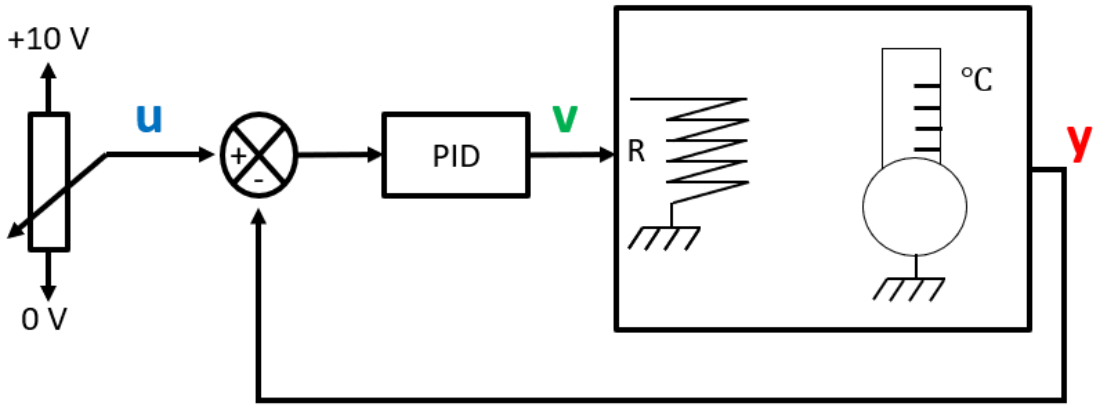


Figura 2.5. Diagrama de bloques del sistema.

Como se puede apreciar en el diagrama de bloques de la **Figura 2.5.** el control del sistema se realizará con un controlador PID, al cual le llegan dos señales una que nos la da el potenciómetro (Temperatura que deseamos tener) y otra que nos la da el sensor (Temperatura real del sistema). El sensor es la realimentación del sistema. Dentro del proceso tenemos una resistencia calorífica que se encenderá o apagará dependiendo de si necesitamos más o menos temperatura en el interior de la caja.

Una vez explicado los componentes del diagrama de bloques de la **Figura 2.5.** se procede a explicar el funcionamiento de todo el conjunto del sistema. Se comienza declarando una temperatura que se quiere obtener en el interior de la caja, para ello tenemos que hacer uso del potenciómetro. Mediante el valor del potenciómetro y el valor que nos da el sensor, que se encuentra en el interior de la caja, se puede calcular el error del sistema (ver ecuación [2.24]). Ese error actúa como entrada para el controlador PID el cual mediante el uso de las ecuaciones [2.25], [2.26] y [2.27] hace una serie de ajustes en el error para tratar de reducirlo y emite una salida que actúa como entrada para la

resistencia calorífica, que se encuentra en el interior de la caja, para que se active y aporte más calor o para que se apague y reduzca la temperatura del interior de la caja. Dentro de la caja el sensor se encarga de recoger la temperatura a la que se encuentra el interior de la caja y nuevamente genera una salida que actúa como entrada para el controlador PID. Todo este proceso es como un bucle infinito que se ejecuta una y otra vez hasta ajustar el error al mínimo.

Para que el controlador PID pueda ajustar el error, las entradas que le llegan (potenciómetro y sensor de temperatura) tienen que estar en las mismas unidades (en este caso voltios), por eso es necesario el uso de un potenciómetro en la entrada que se encarga de convertir la temperatura objetivo que se expresa en grados Celsius a voltios. En próximos capítulos se verá cómo se consigue hacer ese cambio de unidades.

Función que desempeñan cada uno de los componentes del diagrama de bloques de la **Figura 2.5:**

- Potenciómetro: actúa como divisor de tensión variable. Expresa la temperatura deseada (temperatura de referencia) traducida a voltios. Valores entre 0 y 10 V. El potenciómetro es el filtro de entrada porque convierte la variable de entrada (temperatura) a una variable eléctrica para que pueda ser interpretada por el controlador.
- Comparador: calcula el error del sistema, es la diferencia entre la referencia y la salida.
- Controlador PID: se encarga de leer los datos que le llegan y de elaborar una respuesta para el actuador.
- Caja: es la planta (donde se encuentra el proceso y el actuador del sistema), dentro de la caja se encuentra la resistencia calorífica, el sensor de temperatura y el proceso (temperatura interior de la caja).
- Resistencia calorífica: es el actuador, se enciende para aumentar la temperatura del interior de la caja si la temperatura que marca el sensor está por debajo de la temperatura objetivo y se apagará si la temperatura que marca el sensor está por encima de la temperatura objetivo.
- Sensor de temperatura: es la retroalimentación del sistema, mide la temperatura del interior de la caja y expresa el valor en Celsius. Cuando tiene que transmitirle la señal al controlador PID, hace una conversión para pasar grados Celsius a voltios.

2.3 GENERACIÓN DE LA SEÑAL DE REFERENCIA

En este apartado se verá los distintos componentes que se pueden utilizar para generar una señal de referencia y se elegirá uno en concreto que cumpla con las necesidades del proyecto.

2.3.1 Diferentes opciones

Para entender mejor la generación de la señal de referencia, se debe tener en cuenta que la señal de referencia es una variable deseada, es decir, marca el objetivo al que se quiere llegar. Se pueden generar distintos tipos de señales de referencia como podrían ser magnitudes físicas, eléctricas, térmicas, etc. Para facilitarle la tarea al controlador y a nuestro sistema lo mejor será que todas las señales sean eléctricas para poder trabajar con voltios a la hora de simular y representar las señales.

Posibles opciones que actúan en la generación de la señal de referencia:

- Potenciómetro: es una resistencia variable que generalmente se ajusta manualmente. Los potenciómetros utilizan tres terminales y se suelen utilizar en circuitos de poca corriente, para circuitos de mayor corriente se utilizan los reóstatos. En muchos dispositivos eléctricos los potenciómetros son los que establecen el nivel de salida. Por ejemplo, en un altavoz el potenciómetro ajusta el volumen; en una pantalla se puede utilizar para controlar el brillo. Convierten energía mecánica en energía eléctrica.
- Transductor: es un dispositivo que tiene la misión de recibir energía de una naturaleza eléctrica, mecánica, acústica, etc., y suministrar otra energía de diferente naturaleza, pero de características dependientes de la que recibió.
- Selector eléctrico rotativo: tiene la función de abrir o cerrar contactos según una posición seleccionada de manera manual.

En vista de las diferentes opciones que existen para la generación de una señal de referencia, se escoge la de un potenciómetro puesto que permite al usuario seleccionar, de forma sencilla y gráfica, la temperatura deseada, además de que crea una señal eléctrica con la que se puede trabajar en voltios.

2.3.2 Selección del potenciómetro

Una vez visto las diferentes opciones para generar una señal de referencia, se procede a elegir un potenciómetro que sirva para la solución del proyecto.

Primero se debe tener en cuenta que existen diferentes tipos de potenciómetros, y estas clasificaciones dependen de la variación de su resistencia y por el tipo de mando.[16]

Potenciómetro según las variaciones en la resistencia:

- Potenciómetro lineal: Donde la resistencia varia de forma lineal a la variación del ángulo que se hizo variar la resistencia.
- Potenciómetro logarítmico: La resistencia varía en función del ángulo, pero esta vez de forma exponencial inversa.
- Potenciómetro senoidal: Varía en función del seno del ángulo.
- Potenciómetro anti logarítmico: varia exponencialmente al ángulo de giro.

Potenciómetro según el tipo de mando:

- Potenciómetros rotatorios: Se controlan girando un eje. Son de larga duración y ocupan poco espacio.
- Potenciómetro Deslizante: Se controla corriendo un cursor en forma recta. Muy utilizado en ecualizadores gráficos.
- Potenciómetro Múltiple: Tienen los ejes coaxiales. De esta manera ocupan mucho menos espacio.

Para elegir un potenciómetro nos tenemos que fijar en las siguientes características:

- Resistencia máxima que ofrece.
- Tipo de potenciómetro que es (lineal, logarítmico, etc.)

Una vez visto que es un potenciómetro y que características en concreto se necesitan, pasamos a elegir uno que sirva para el caso en particular. Se escoge el potenciómetro B10K [17], el cual tiene las siguientes características:

- Resistencia máxima que ofrece: 10 kΩ.
- Tipo de potenciómetro: lineal.

En caso de necesitar alguna característica más específica podemos acceder a la ficha técnica del potenciómetro: [17]

Hay que tener en cuenta que un potenciómetro puede funcionar como divisor de voltaje o como resistencia variable. Para el proyecto se utiliza como divisor de voltaje. También se debe tener en cuenta que se puede escoger muchas más variantes para el potenciómetro, es decir, que en vez de

escoger una resistencia máxima de 10 kΩ se puede coger uno que tenga una resistencia máxima de 100 kΩ.

Como observación: la letra 'B' del potenciómetro B10k quiere decir que el potenciómetro es de tipo lineal ya que en el mercado existen potenciómetros logarítmicos, aunque los más comunes son los lineales.

Figura 2.6. potenciómetro B10K. [17]

Se debe tener en cuenta que a la hora de representar las señales de entrada del sistema (u), salida del controlador PID (v) y la salida del sistema (y), se necesita tenerlo todo en voltios para poder representarlo. La entrada del sistema inicialmente se da en grados Celsius, pero al utilizar un potenciómetro se pasan los grados a voltios. Para hacer esto en la aplicación se necesita aplicar una ecuación de la recta (porque el potenciómetro es lineal), donde se sabe que el valor más pequeño para la temperatura será 0 ℃ y que el máximo valor es 50 ℃. Lo mismo para los voltios, el valor mínimo es 0 V y el máximo 10 V. Ecuación de la recta [2.11]:

$$y - y_1 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right) \cdot (x - x_1) \qquad [2.11]$$

Para el caso, los valores de 'x', son los de la temperatura, los valores de 'y' son los voltios. Por tanto, para 0 grados de temperatura se tiene 0 voltios y para 50 grados de temperatura se tiene 10 voltios:

$$(x_1 = 0,\ y_1 = 0)$$

$$(x_2 = 50,\ y_2 = 10)$$

Sustituyendo los puntos en la ecuación [2.11]:

$$y - 0 = \left(\frac{0 - 10}{0 - 50}\right) \cdot (x - 0) \rightarrow y = \frac{1}{5} \cdot (x) \qquad [2.12]$$

La ecuación [2.12] sirve para pasar los grados Celsius a voltios y será la ecuación del potenciómetro.

2.4 COMPARADOR

En este apartado se verá los distintos componentes que se pueden utilizar para comparar señales y se elegirá uno en concreto que cumpla con las necesidades del proyecto.

2.4.1 Diferentes opciones

Los comparadores realizan operaciones matemáticas simples como suma, resta, multiplicación y algunas veces combinaciones de éstas.

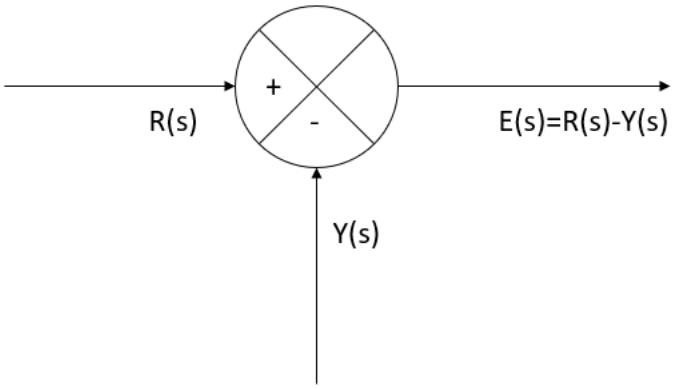


Figura 2.7. Comparador con operación matemática resta.

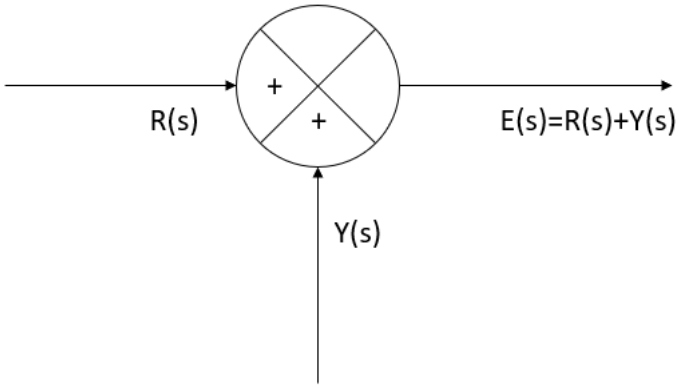


Figura 2.8. Comparador con operación matemática suma.

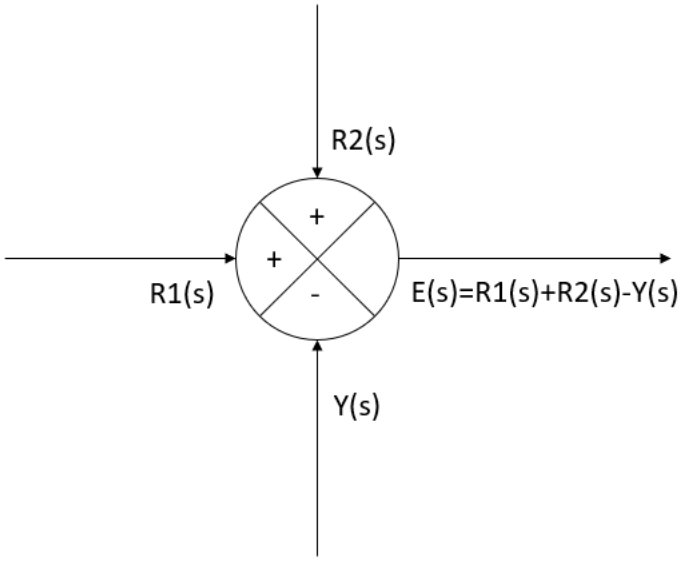


Figura 2.9. Comparador con operación matemática suma y resta.

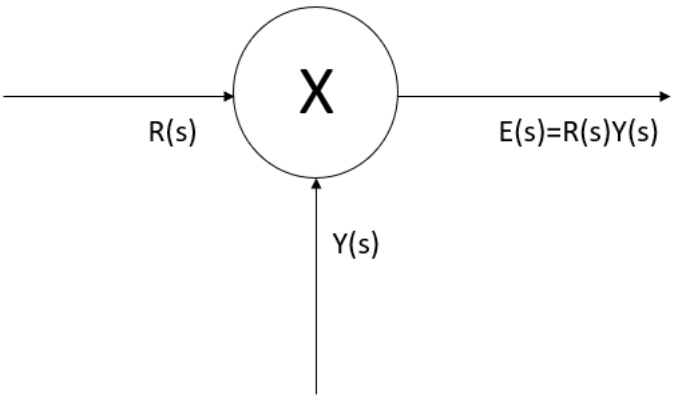


Figura 2.10. Comparador con operación matemática multiplicación.

2.4.2 Selección del comparador

En vista de las diferentes opciones de comparador el que más encaja con el proyecto, es el comparador con la operación matemática resta (ver **Figura 2.7**) ya que se necesita conocer la diferencia que existe entre la señal de referencia y la señal de salida para que el regulador pueda hacer los ajustes necesarios para reducir el error al mínimo.

Una vez elegido el comparador se puede calcular el error del sistema. El error del sistema es la diferencia entre la entrada de referencia (u) y la salida del sistema (y). Se necesita tenerlo todo en voltios para poder representarlo. Ver ecuación [2.13]:

$$error = u - y \qquad [2.13]$$

2.5 REGULADOR PID

En este apartado se verá los distintos componentes que se pueden utilizar para regular señales y se elegirá uno en concreto que cumpla con las necesidades del proyecto.

2.5.1 Diferentes opciones

Un controlador automático compara el valor real de la salida de una planta con la entrada de referencia (el valor deseado), determina la desviación y produce una señal de control que reducirá la desviación a cero o a un valor pequeño. La manera en la cual el controlador automático produce la señal de control se denomina acción de control. El controlador detecta la señal de error que, por lo general, está en un nivel de potencia muy bajo, y la amplifica a un nivel lo suficientemente alto. La salida de un controlador automático se alimenta a un actuador, tal como un motor neumático, una válvula, un motor hidráulico o un motor eléctrico. [15]

Clasificación de los controladores industriales. Los controladores industriales se clasifican, de acuerdo con sus acciones de control, como:

- De dos posiciones o de encendido y apagado (on / off): el elemento de actuación sólo tiene dos posiciones, encendido y apagado. El control de dos posiciones o de encendido y apagado es relativamente simple y barato, razón por la cual su uso es extendido en sistemas de control tanto industriales como domésticos.
- Proporcionales.
- Integrales.
- Proporcionales-integrales.
- Proporcionales-derivativos.
- Proporcionales-integrales-derivativos.

Casi todos los controladores industriales emplean como fuente de energía la electricidad o un fluido presurizado, tal como el aceite o el aire. Los controladores también pueden clasificarse, de acuerdo con el tipo de energía que utilizan en su operación, como neumáticos, hidráulicos o electrónicos. El tipo de controlador que se use debe decidirse con base en la naturaleza de la planta y las condiciones operacionales, incluyendo consideraciones tales como seguridad, costo, disponibilidad, confiabilidad, precisión, peso y tamaño.[15]

Para un controlador con acción de control proporcional, la relación entre la salida del controlador u(t) y la señal de error e(t) es:

$$u(t) = K_p \cdot e(t) \quad [2.14]$$

Aplicando la transformada de Laplace a la ecuación [2.14]:

$$\frac{U(s)}{E(s)} = K_p \quad [2.15]$$

En donde K_p se considera la ganancia proporcional. El controlador proporcional es, en esencia, un amplificador con una ganancia ajustable.

En un controlador con acción de control integral, el valor de salida del controlador u(t) se cambia a una razón proporcional a la señal de error e(t). Es decir:

$$\frac{du(t)}{dt} = K_i \cdot e(t) \quad [2.16]$$

Aplicando la transformada de Laplace a la ecuación [2.16]:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \quad [2.17]$$

Si se duplica el valor de e(t), el valor de u(t) varía dos veces más rápido. Para un error de cero, el valor de u(t) permanece estacionario. En ocasiones, la acción de control integral se denomina control de reajuste (reset). [15]

La acción de control de un controlador proporcional-integral (PI) se define mediante:

$$u(t) = K_p \cdot e(t) + \frac{K_p}{T_i}\int_0^t e(t)\,dt \quad [2.18]$$

Aplicando la transformada de Laplace a la ecuación [2.18]:

$$\frac{U(s)}{E(s)} = K_p \cdot \left(1 + \frac{1}{T_i \cdot s}\right) \quad [2.19]$$

En donde K_p es la ganancia proporcional y T_i se denomina tiempo integral. Tanto K_p como T_i son ajustables. El tiempo integral ajusta la acción de control integral, mientras que un cambio en el valor de K_p afecta las partes integral y proporcional de la acción de control.[15]

La acción de control de un controlador proporcional-derivativa (PD) se define mediante:

$$u(t) = K_p \cdot e(t) + K_p \cdot T_d \frac{de(t)}{dt} \quad [2.20]$$

Aplicando la transformada de Laplace a la ecuación [2.20]:

$$\frac{U(s)}{E(s)} = K_p \cdot (1 + T_d \cdot s) \quad [2.21]$$

En donde K_p es la ganancia proporcional y T_d es una constante denominada tiempo derivativo. Tanto K_p como T_d son ajustables. La acción de control derivativa, en ocasiones denominada control de velocidad, ocurre donde la magnitud de la salida del controlador es proporcional a la velocidad de cambio de la señal de error. El tiempo derivativo T_d es el intervalo de tiempo durante el cual la acción de la velocidad hace avanzar el efecto de la acción de control proporcional. Aunque la acción de control derivativa tiene la ventaja de ser de previsión, tiene las desventajas de que amplifica las señales de ruido y puede provocar un efecto de saturación en el actuador. Observe que la acción de control derivativa no se usa nunca sola, debido a que sólo es eficaz durante periodos transitorios.[15]

La combinación de una acción de control proporcional, una acción de control integral y una acción de control derivativa se denomina acción de control proporcional-integral-derivativa (PID). Esta acción combinada tiene las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:

$$u(t) = K_p \cdot e(t) + \frac{K_p}{T_i} \int_0^t e(t)\, dt + K_p \cdot T_d \frac{de(t)}{dt} \quad [2.22]$$

Aplicando la transformada de Laplace a la ecuación [2.22]:

$$\frac{U(s)}{E(s)} = K_p \cdot \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right) \quad [2.23]$$

En donde K_p es la ganancia proporcional, T_i es el tiempo integral y T_d es el tiempo derivativo. [15]

El valor proporcional depende del error actual, el integral depende de los errores pasados y el derivativo es una predicción de los errores futuros. Al usar las tres acciones a la vez se puede ajustar el control de un proceso para que, por ejemplo: se aporte más calor, se abra más una válvula, se aumente la velocidad de llenado, etc.

2.5.2 Selección del regulador

Los tipos de controladores disponibles para el diseño de sistemas de control están limitados sólo por la imaginación. Los ingenieros prácticos normalmente establecen que uno escoge el controlador más simple que cumpla con todas las especificaciones de diseño. En la mayoría de los casos, mientras más complejo sea un controlador, es más costoso, menos confiable, y más difícil de diseñar. El escoger un controlador determinado para una aplicación específica se basa a menudo en la experiencia del diseñador, y algunas veces en la intuición, e involucra inevitablemente tanto arte como ciencia.[13]

Para elegir un controlador PID para el proyecto se tendrán en cuenta las siguientes ecuaciones de un PID discreto:

$$error = entrada - salida \quad [2.24]$$

$$P = kp \cdot error \quad [2.25]$$

$$D = \frac{kd \cdot (error_{actual} - error_{pasado})}{Ts} \quad [2.26]$$

$$I_{actual} = I_{anterior} + \frac{ki \cdot T_s \cdot (error_{pasado} + error_{actual})}{2} \quad [2.27]$$

Con las ecuaciones del PID [2.24] [2.25] [2.26] [2.27] se puede corregir el error del sistema por lo que se estará simulando el funcionamiento de un controlador PID. En la **Figura 2.11.** se puede observar el aspecto que tiene un controlador PID comercial.

Figura 2.11. Controlador PID comercial.[18]

Para simular el funcionamiento de un controlador PID, sólo se utilizarán las ecuaciones ***[2.24] [2.25] [2.26] y [2.27]***, por lo que no tiene sentido que se busquen especificaciones de un controlador PID.

2.6 ACTUADOR

En este apartado se verá los distintos actuadores que se pueden utilizar para la planta y se elegirá uno en concreto que cumpla con las necesidades del proyecto.

2.6.1 Diferentes opciones

El actuador es un dispositivo de potencia que produce la entrada para la planta de acuerdo con la señal de control, a fin de que la señal de salida se aproxime a la señal de entrada de referencia. La salida de un controlador automático se alimenta a un actuador, tal como un motor neumático, una válvula, un motor hidráulico, un motor eléctrico, una resistencia, etc. [15]

Existen muchos tipos de actuadores, pero para el proyecto sólo se necesitan aquellos que sean capaces de producir calor a partir de un voltaje, por lo que se tiene las siguientes opciones:

- Célula Peltier: es un dispositivo que mediante el paso de corriente a través de su circuito es capaz de refrigerar por un lado y calentar por el otro.
- Resistencias calefactoras: son aparatos que calientan o aumentan la temperatura, transforman la energía eléctrica en calor. La gran mayoría de ellas son fabricadas con un alambre de aleación de níquel y cromo. Esta aleación soporta temperaturas muy altas (de hasta 1000 ℃ o más).

2.6.2 Selección del actuador

En vista de las diferentes opciones de actuadores, que actúan como elemento calefactor, el que más encaja con el proyecto, es el de resistencias calefactoras ya que la célula Peltier es bastante ineficiente a la hora de calentar y se suelen utilizar más para refrigerar. Para seleccionar una resistencia que sirva como calefactor se optará por resistencias de nicromo. El nicromo es una aleación de níquel (80%) y cromo (20%). Destaca por ser un metal muy resistente a las altas temperaturas y por tener una elevada resistencia eléctrica. El nicromo posee las siguientes propiedades:

- Alta resistencia a la corrosión y la oxidación.
- Resistencia a las altas temperaturas (punto de fusión elevado).
- Elevada resistencia eléctrica.
- No es magnético.
- Color gris plateado.
- Resistente y flexible.

El nicromo se suele utilizar en secadores de pelo, hornos eléctricos, elementos de calefacción eléctricos, etc. El nicromo se suele vender en forma de alambre con distintos diámetros y longitud, por lo que da la opción al consumidor de poder crear las resistencias calefactoras a su gusto.

Para elegir una resistencia calorífica de nicromo se debe tener en cuenta las siguientes características:

- Resistencia por unidad de longitud.
- Diámetro del alambre.
- Longitud que se quiere utilizar.
- Temperatura que puede alcanzar sin perder sus propiedades como resistencia.
- Material del que está formado.
- Dimensiones.
- Masa de la resistencia calorífica.

Una vez visto que es una resistencia calorífica de nicromo y que características en concreto se necesitan, se elige una que sirva. Se escoge la "RS PRO-TEST Lead Wire Single Core 36m", la cual tiene las siguientes características:

- Resistencia por unidad de longitud: 11.486 Ω/m
- Diámetro del alambre: 0.345 mm
- Longitud que se quiere utilizar: 0.2 m
- Temperatura que puede alcanzar sin perder sus propiedades como resistencia: 1150 ℃
- Material del que está formado: nicromo (Ni-Cr).
- Dimensiones: [1 cm de diámetro x 4.5 cm de largo]
- Masa de la resistencia calorífica: 4 gr

En caso de necesitar alguna característica más específica se puede acceder a la ficha técnica de la resistencia la "RS PRO-TEST Lead Wire Single Core 36m": [19]

Figura 2.12. Resistencia calorífica de nicromo, "RS PRO-TEST Lead Wire Single Core 36m".[19]

Según la longitud del alambre de nicromo, su diámetro y resistencia por unidad de longitud se puede calcular la resistencia que se tendrá para el proyecto, según la siguiente ecuación:

$$R = \rho \cdot \frac{l}{A} \qquad [2.28]$$

Despejando la ecuación [2.28] para obtener la resistencia por unidad de longitud:

$$\frac{R}{l} = \frac{\rho}{A} \qquad [2.29]$$

Como en el datasheet de la resistencia calorífica "RS PRO-TEST Lead Wire Single Core 36m" se da el valor de la resistencia por unidad de longitud ($11.486\ \Omega/m$) simplemente se tiene que sustituir el valor en la ecuación [2.29] y además se debe tener en cuenta que la longitud de la resistencia que se quiere obtener es de 0.2 m. Sustituyendo valores en la ecuación [2.29]:

$$\frac{R}{0.2\ m} = 11.486 \frac{\Omega}{m} \rightarrow R = 2.3\ \Omega$$

Se puede observar que para esos valores del nicromo se obtiene una resistencia de 2.3 Ω. Gracias a las propiedades del nicromo la resistencia permanecerá constante, sin que se vea influenciada por el incremento de temperatura.

2.7 SENSOR

En este apartado se verá los distintos sensores que se pueden utilizar para detectar la señal a la salida del sistema y se elegirá uno en concreto que cumpla con las necesidades del proyecto.

2.7.1 Diferentes opciones

El sensor, o elemento de medición, es un dispositivo que convierte la variable de salida en otra variable manejable, tal como un desplazamiento, una presión, o un voltaje, que pueda usarse para comparar la salida con la señal de entrada de referencia. Este elemento está en la trayectoria de realimentación del sistema en lazo cerrado. El punto de ajuste del controlador debe convertirse en una entrada de referencia con las mismas unidades que la señal de realimentación del sensor o del elemento de medición.[15]

En función de lo que se quiere medir, existen muchos tipos de sensores diferentes:

- Mediciones eléctricas:
    - Tensión, corriente, potencia, conductividad, etc.
- Mediciones térmicas:
    - Termistores, termopares, pirómetros (laser, infrarrojos, ópticos).
- Mediciones mecánicas:
    - Desplazamiento, posición, distancia y nivel, transductores de fuerza, transductores de movimiento (velocidad y aceleración).
- Mediciones en fluidos y gases:
    - Transductores de presión, de flujo y caudal.
- Mediciones en procesos químicos:
    - PH, concentraciones, detección de gases, humedad, etc.

Existen muchos tipos de sensores, pero para el proyecto sólo se necesitan aquellos que sean capaces de medir la temperatura del interior de una caja y emitir una señal en voltios para el controlador PID, por lo que se tiene que elegir entre los siguientes sensores de temperatura:

- Termocuplas o termopares: son sensores muy utilizados en zonas industriales. Está formado por dos materiales distintos unidos en un extremo, al aplicar temperatura en la unión de los metales se genera una variación de voltaje que sirve para medir la temperatura.

- Termistores: su funcionamiento se basa en la variación de la resistividad que presenta un semiconductor con la variación de la temperatura. Existen dos tipos: NTC y PTC. Los NTC son resistencias de coeficiente de temperatura negativa. Los PTC son resistencias con coeficientes de temperatura positiva.
- Sensor de temperatura por infrarrojos: miden la temperatura de la superficie mediante la conversión de energía térmica radiada desde cualquier superficie en su campo de visión en una señal eléctrica con un tiempo de respuesta inferior a un segundo.

2.7.2 Selección del sensor

En vista de las diferentes opciones de sensores de temperatura el que más encaja con el proyecto, es aquel que tenga una salida analógica y que se pueda medir en voltios. Si la salida del sensor de temperatura está dada en voltios se podrá calcular el error del sistema con facilidad porque tanto la salida como la entrada del sistema estarán en las mismas unidades. Para elegir un sensor de temperatura, se deben tener en cuenta las siguientes características:

- Tensión que admite el sensor.
- Rango de temperaturas que puede medir.
- Tipo de sensor (digital o analógico).
- Dimensiones (tamaño del sensor).
- Resolución por cada grado centígrado.
- Unidades de la salida del sensor.

Una vez visto que es un sensor de temperatura y que características en concreto se necesitan, se pasa a elegir un sensor que tenga esas características y que sirva para el caso en particular. Se escoge el sensor LM35, el cual tiene las siguientes características:

- Tensión que admite: [de 4 V a 30 V]
- Rango de temperaturas que puede medir: [-55 ℃ a 150 ℃]
- Tipo de sensor: analógico.
- Dimensiones: [6 mm de diámetro x 30 mm de largo]
- Resolución por cada grado centígrado: 10 mV por cada grado Celsius.

- Unidades de la salida del sensor: en mV.

El sensor LM35 posee una salida analógica en milivoltios y es lineal por lo que facilita los cálculos. En caso de necesitar alguna característica más específica se puede acceder a la ficha técnica del sensor LM35: [20]

Figura 2.13. Sensor de temperatura LM35.[20]

Se debe tener en cuenta que a la hora de representar las señales de entrada del sistema (u), salida del controlador PID (v) y la salida del sistema (y), se necesita tenerlo todo en voltios para poder representarlo. Se ha conseguido representar la entrada del sistema (u) en voltios, pero también se debe representar la salida del sistema (y) en voltios. El sensor de temperatura LM35 ofrece los resultados en milivoltios, siendo el valor más pequeño que puede ofrecer -550 mV para una temperatura de -55 ℃ y un valor máximo de 1500 mV para una temperatura de 150 ℃. En el proyecto, no se llega a temperaturas tan bajas y ni tan altas porque el valor mínimo, para la temperatura es de 0 ℃ y para la tensión es de 0 V, por esta parte no hay problema porque el sensor LM35 arroja un valor de 0 V para una temperatura de 0 ℃. En el proyecto la temperatura más alta que se representará será un valor de 50 ℃ que equivale a 10 V de tención, en esta parte surge el problema porque el valor de tensión que arroja el sensor LM35 para 50 ℃ es de 500 mV, o lo que es lo mismo 0.5 V. Este valor es bastante más pequeño que el de 10 V. Para solucionar esto se opta por utilizar una ganancia a la salida del sensor para hacer más grande el valor de 0.5 V. Si se utiliza una ganancia de 20 se obtendrá el valor de 10 V a la salida del sensor ($20 \cdot 0.5 = 10\ V$).

2.8 PROCESO

En este apartado se hablará sobre las ecuaciones diferenciales de la transferencia de calor entre los diferentes elementos del sistema, dichas ecuaciones servirán para implementarse en el software.

2.8.1 Caja

Antes de mencionar las ecuaciones es conveniente hacer una mención a la caja que contendrá los componentes del sistema, ya que será de gran utilidad para los cálculos.

La caja es un recipiente de diferentes materiales, tamaños y formas, generalmente con tapa que sirve para guardar o transportar cosas. Para elegir una caja se deben tener en cuenta las siguientes características:

- Dimensiones de la caja.
- Material del que está formada la caja.

Una vez visto que es una caja y que características en concreto se necesitan, pasamos a elegir una que sirva para nuestro caso en particular:

- Dimensiones de la caja: [150x150x150 mm]
- Material del que está formada la caja: madera.

Para elegir las dimensiones de la caja se debe tener en cuenta que en el interior de la caja sólo estará el sensor y la resistencia calorífica, como se sabe el tamaño de estos podremos confirmar que con una caja de esas dimensiones entrará el sensor y la resistencia calorífica sin problemas.

2.8.2 Ecuaciones diferenciales

Una vez visto las características de la caja y los componentes que estarán en su interior, se puede hablar de las ecuaciones en diferencias que se necesitan para simular el funcionamiento de estos componentes de forma virtual.

Primero se hablará sobre la medición de la temperatura interior de la caja, es decir, sobre la temperatura que recogerá el sensor. Para ello se debe tener en cuenta las medidas de la caja (150x150x150 mm), del sensor (6 mm de diámetro x 30 mm de largo) y de la resistencia calorífica (1 cm de diámetro x 4.5 cm de largo). Situando la resistencia en el centro de la base de la caja con una orientación vertical y situando el sensor en el centro de la base superior de la caja con una orientación

vertical (ver **Figura 2.14**), se puede apreciar que existe un espacio de aire entre la resistencia y el sensor. Con estas observaciones podemos sacar la conclusión de que existen tres tipos de temperatura:

- Temperatura de la resistencia calorífica (Tc).
- Temperatura del aire de la caja (Th).
- Temperatura del sensor (Tm).

Como se puede apreciar en la **Figura 2.14** el volumen de aire del interior de la caja será mayor que el volumen que ocupa la resistencia calorífica, por lo que, a la hora de hacer mediciones con el sensor de temperatura, un porcentaje grande (95%) irá destinado a medir la temperatura del aire interior de la caja y otro a medir la temperatura de la resistencia calorífica (5%). Con estos datos se puede obtener la primera ecuación:

$$T_m = 0.95 \cdot T_h + 0.05 \cdot T_c \qquad [2.30]$$

Las unidades que se obtienen en la ecuación [4.1] son Celsius (°C).

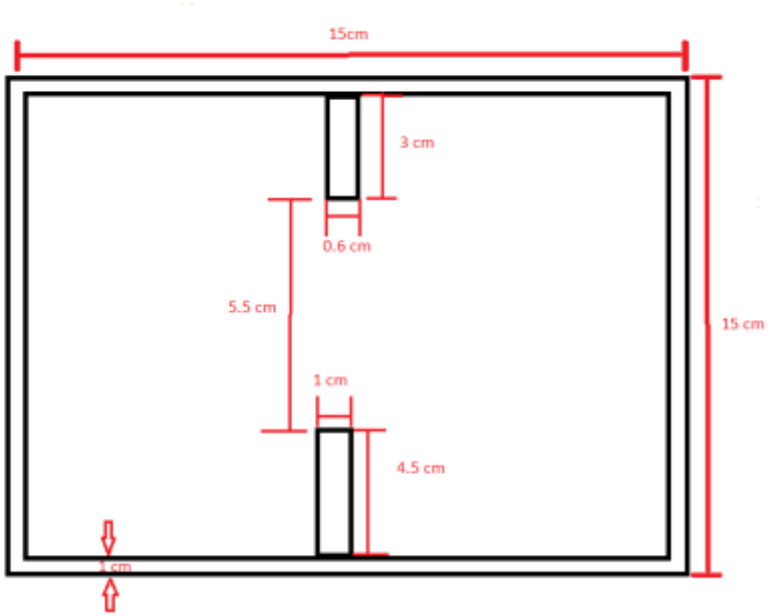


Figura 2.14. Boceto del interior de la caja. En la parte inferior se representa a la resistencia calorífica y en la parte superior se representa al sensor de temperatura.

Una vez visto la obtención de la temperatura del sensor (Tm), se debe hablar de las entradas que le daremos al controlador PID para que pueda corregir el error del sistema. ¿Cómo se calcula el error? El error se calcula restando la temperatura de referencia (Tr) menos la temperatura de salida del

sistema, que es la temperatura del sensor (Tm). Dependiendo de los ajustes que se quiera hacer para corregir el error, se puede utilizar la parte proporcional (P), integral (I) o derivativa (D):

$$error = T_r - T_m \quad [2.31]$$

$$P = kp \cdot error \quad [2.32]$$

$$D = \frac{kd \cdot (error_{actual} - error_{pasado})}{Ts} \quad [2.33]$$

$$I_{actual} = I_{anterior} + \frac{ki \cdot T_s \cdot (error_{pasado} + error_{actual})}{2} \quad [2.34]$$

Las ecuaciones [2.32], [2.33] y [2.34] son las ecuaciones del PID por separado y se pueden utilizar individualmente, pero si se quiere utilizar dos de los componentes del controlador PID, por ejemplo, el PI, se deberá sumar la ecuación [2.32] con la ecuación [2.34], y si se quiere utilizar todas las ecuaciones del PID se sumarán las ecuaciones [2.32], [2.33] y [2.34].

Siguiendo un orden lógico toca hablar sobre la energía calorífica (Q) que generará la resistencia calorífica de nicromo según la entrada que le entregue el controlador PID. Pero antes se debe recordar cual es el valor de la resistencia (ver ecuación [2.29]).

$$R = 2.3\ \Omega$$

La ecuación de la energía calorífica para el sistema será:

$$Q = 0.24 \cdot \frac{V^2}{R} \quad [2.35]$$

En la ecuación [2.35], la (V) es la tensión que devuelven las ecuaciones del PID, la (R) es el valor de la resistencia calorífica de nicromo y los (0.24) sirven para pasar los vatios a calorías entre segundo. Las unidades que se obtienen en la ecuación [2.35] son calorías entre segundos (cal/s).

Llegados a este punto se tiene que calcular la variación de la temperatura de la resistencia calorífica en función del tiempo ($\partial T_c / \partial t$):

$$M_c \cdot \frac{\partial T_c}{\partial t} = Q - U_c \cdot S_c \cdot (T_c - T_h) \quad [2.36]$$

Despejando (Mc) se obtiene:

$$\frac{\partial T_c}{\partial t} = \frac{Q - U_c \cdot S_c \cdot (T_c - T_h)}{M_c} \quad [2.37]$$

Se calcula la capacidad calorífica de la resistencia calorífica, que se representará como (Mc). La capacidad calorífica (Mc) es el cociente del calor especifico (c) por la masa de la sustancia (m). El calor especifico es la cantidad de calor necesario para elevar la temperatura de una unidad de masa de una sustancia en un grado. Revisando las características de la resistencia de nicromo se obtiene que el material que utiliza esta resistencia es nicromo (Ni-Cr), por lo que su calor específico (c) es 444 ($\frac{J}{kg \cdot ℃}$), valor que se obtiene de la tabla [21] y que su masa (m) es de 0.004 Kg.

$$M_c = c \cdot m \qquad [2.38]$$

Se sustituyen los valores en la ecuación [2.38]:

$$M_c = 444 \cdot 0.004 = 1.776 \frac{J}{℃}$$

Para tener todo en la misma unidad se debe pasar los julios a calorías:

$$M_c = 1.776 \ \frac{J}{℃} \cdot 0.24 \ \frac{cal}{J} = 0.42624 \ \frac{cal}{℃}$$

Se necesita conocer la transmitancia térmica de la resistencia calorífica (Uc). La transmitancia térmica, es la medida del calor que fluye por unidad de tiempo y superficie, transferido a través de un sistema constructivo, formado por una o más capas de material.

$$U_c = \frac{1}{Rt_{nicromo}} \qquad [2.39]$$

Para este apartado solo se tiene una capa de material, el nicromo de la resistencia calorífica. Para calcular la resistencia térmica del nicromo se necesita conocer su espesor y su conductividad térmica (λ). En la **Figura 2.14** se puede observar que la resistencia tiene un espesor de 1 cm. Para obtener el valor de la conductividad térmica se usará la tabla [22], en la que se obtiene un valor de conductividad térmica para el nicromo de 11.3 ($\frac{W}{m \cdot ℃}$). Con estos valores se puede usar la siguiente ecuación:

$$Rt_{nicromo} = \frac{espesor}{\lambda} \qquad [2.40]$$

Sustituyendo los valores en [2.40]:

$$Rt_{nicromo} = \frac{0.01}{11.3} = 8.85 \cdot 10^{-4} \ \frac{m^2 \cdot ℃}{W}$$

Sustituyendo los valores de [2.40] en [2.39]:

$$U_c = \frac{1}{8.85 \cdot 10^{-4}} = 1129.94 \ \frac{W}{m^2 \cdot ℃}$$

Se necesita conocer la superficie de contacto de la resistencia calorífica (Sc). La superficie de contacto de la resistencia calorífica (Sc), será el área. Área de un cilindro [2.41]:

$$S_c = 2 \cdot \pi \cdot r \cdot h \qquad [2.41]$$

Dimensiones de la resistencia (1 cm de diámetro x 4.5 cm de largo). Se pasan los centímetros a metros y se sustituyen los valores en la ecuación [2.41]:

$$S_c = 2 \cdot \pi \cdot r \cdot h = 2 \cdot \pi \cdot 0.5 \cdot 10^{-2} \cdot 4.5 \cdot 10^{-2} = 1.4137 \cdot 10^{-3} \ m^2$$

Como (Uc) y (Sc) son valores constantes y en la ecuación [2.37] (Uc) aparece multiplicando a (Sc), se hace la operación para ver qué valor sale:

$$U_c \cdot S_c = 1129.94 \cdot 1.4137 \cdot 10^{-3} = 1.5974 \ \frac{W}{℃} \cdot 0.24 = 0.3834 \ \frac{cal}{s \cdot ℃}$$

Se debe tener en cuenta que 1 vatio es igual a 1 julio entre segundo y que 1 julio son 0.24 calorías.

Por último, se sustituye los nuevos valores que se obtienen en la ecuación [2.37]:

$$\frac{\partial T_c}{\partial t} = \frac{Q - U_c \cdot S_c \cdot (T_c - T_h)}{M_c} = \frac{Q - 0.3834 \cdot (T_c - T_h)}{0.42624}$$

Llegados a este punto se observa que ciertos valores de la ecuación [2.37] serán constantes (Uc, Sc, Mc) y otros serán variables (Tc, Q, Th). Las unidades que se obtiene en la ecuación [2.37] son grados Celsius entre segundos ($℃/s$).

Falta calcular la variación de la temperatura del aire en el interior de la caja en función del tiempo ($\partial T_h / \partial t$):

$$M_h \cdot \frac{\partial T_h}{\partial t} = U_c \cdot S_c \cdot (T_c - T_h) - U_p \cdot S_p \cdot (T_h - T_e) \qquad [2.42]$$

Despejando (Mh) obtenemos:

$$\frac{\partial T_h}{\partial t} = \frac{U_c \cdot S_c \cdot (T_c - T_h) - U_p \cdot S_p \cdot (T_h - T_e)}{M_h} \qquad [2.43]$$

Se calcula la capacidad calorífica del aire que se representa como (Mh). En primer lugar, ¿cuál es la masa de aire que existe en el interior de la caja? para ello se necesita conocer el volumen del recipiente que ocupa (el volumen de la caja = 15x15x15 cm) y conocer cuál es la densidad del aire a 0 ℃ (1.29 kg/m^3), este valor se obtiene de la tabla [23]. Aplicando la ecuación para el cálculo de la masa [2.44]:

$$m = Volumen \cdot Densidad \quad [2.44]$$

Se pasa el volumen de centímetros cúbicos a metros cúbicos para que las unidades estén en el sistema internacional. sustituyendo los valores en la ecuación [2.44]:

$$m = 3.375 \cdot 10^{-3} \cdot 1.29 = 4.35375 \cdot 10^{-3}\ kg$$

Una vez conocida la cantidad de masa existente en el interior de la caja, se necesita saber el valor del calor especifico del aire, que es 1000 ($J/kg \cdot k$), este valor se obtiene de la tabla [23]. Se define la ecuación para calcular la capacidad calorífica del aire:

$$M_h = c \cdot m \quad [2.45]$$

Sustituyendo los valores en la ecuación [2.45]:

$$M_h = 1000 \cdot 4.35375 \cdot 10^{-3} = 4.35375 \frac{J}{℃}$$

Para tener todo en la misma unidad se debe pasar los julios a calorías:

$$M_h = 4.35375\ \frac{J}{℃} \cdot 0.24\ \frac{cal}{J} = 1.0449 \frac{cal}{℃}$$

Se necesita conocer la transmitancia térmica del aire interior y la pared de madera de la caja (Up).

$$U_p = \frac{1}{Rt_{aire} + Rt_{madera}} \quad [2.46]$$

Para este apartado se tiene dos capas de material, el aire del interior y la pared de la caja. Para calcular la resistencia térmica del aire interior se necesita conocer su espesor y su conductividad térmica (λ) y lo mismo para la pared de la caja. En la **Figura 2.14** se observa que el aire interior tiene un espesor de 13 cm y que la pared de la caja tiene un espesor de 1 cm. Para obtener el valor de la conductividad térmica se usará la tabla [24], en la que se obtiene un valor de conductividad térmica del aire de 0.02 ($\frac{W}{m \cdot ℃}$) y para la pared de madera de la caja un valor de 0.13 ($\frac{W}{m \cdot ℃}$). Se usa la misma ecuación que se utilizó para el nicromo [2.40]:

$$Rt_{aire} = \frac{0.13}{0.02} = 6.5\ \frac{m^2 \cdot ℃}{W}$$

$$Rt_{madera} = \frac{0.01}{0.13} = 0.077\ \frac{m^2 \cdot ℃}{W}$$

Sustituyendo en [2.46]:

$$U_p = \frac{1}{6.5 + 0.077} = 0.152 \ \frac{W}{m^2 \cdot ℃}$$

Se necesita conocer la superficie de contacto del aire con las paredes de la caja (Sp), eso es el área de las paredes. Como un cubo tiene 6 caras se multiplica al área de un cuadrado por 6:

$$S_p = 6 \cdot L \cdot L \qquad [2.47]$$

Se pasa los centímetros a metros y se sustituyen los valores en la ecuación [2.47]:

$$S_p = 6 \cdot L \cdot L = 6 \cdot 0.15 \cdot 0.15 = 0.135 \ m^2$$

Como (Up) y (Sp) son valores constantes y en la ecuación [2.43] (Up) aparece multiplicando a (Sp), se hace la operación para ver qué valor sale:

$$U_p \cdot S_p = 0.152 \cdot 0.135 = 0.02052 \frac{W}{℃} \cdot 0.24 = 4.9248 \cdot 10^{-3} \ \frac{cal}{s \cdot ℃}$$

Se debe tener en cuenta que 1 vatio es igual a 1 julio entre segundo y que 1 julio son 0.24 calorías.

Por último, se sustituyen los nuevos valores que se han obtenido en la ecuación [2.43]:

$$\frac{\partial T_h}{\partial t} = \frac{0.3834 \cdot (T_c - T_h) - 4.9248 \cdot 10^{-3} \cdot (T_h - T_e)}{1.0449}$$

Llegados a este punto se puede observar que se tiene ciertos valores de la ecuación [2.43] que serán constantes (Uc, Sc, Mh, Up, Sp) y otros que serán variables (Tc, Th, Te). El valor (Te) será la temperatura exterior de la caja, este valor actuará como perturbación. Las unidades que se obtienen en la ecuación [2.43] son grados Celsius entre segundos ($℃/s$).

3 IMPLEMENTACIÓN DEL SOFTWARE

En este apartado se verá el planteamiento que se ha seguido para convertir las caracteristicas reales en virtuales para que el software simule lo más parecido a la realidad. Se verá la interfaz gráfica de usuario con sus principales funciones y se verá el código del software.

3.1 DISCRETIZACIÓN DE LAS ECUACIONES DIFERENCIALES

Para poder simular el funcionamiento de una planta de control de temperatura de forma virtual, se necesita recordar cuales son las ecuaciones más importantes que tratan de replicar el funcionamiento de los componentes de la planta:

$$T_m = 0.95 \cdot T_h + 0.05 \cdot T_c \quad [2.30]$$

$$Q = 0.24 \cdot \frac{V^2}{R} \quad [2.35]$$

$$\frac{\partial T_c}{\partial t} = \frac{Q - U_c \cdot S_c \cdot (T_c - T_h)}{M_c} \quad [2.37]$$

$$\frac{\partial T_h}{\partial t} = \frac{U_c \cdot S_c \cdot (T_c - T_h) - U_p \cdot S_p \cdot (T_h - T_e)}{M_h} \quad [2.43]$$

La ecuación 2.30 da los valores que arroja el sensor, la ecuación 2.35 muestra los valores que obtiene la resistencia calorífica, estas ecuaciones no hay que discretizarlas porque no tienen derivadas en función del tiempo.

La ecuación 2.37 da los valores de temperatura que alcanza la resistencia calorífica y la ecuación 2.43 muestra los valores de temperatura que se alcanza en el interior de la caja en función del tiempo. Como estas ecuaciones poseen derivadas, se tienen que discretizar:

$$\frac{\partial T_c}{\partial t} = \frac{T_{c_{actual}} - T_{c_{anterior}}}{t_s} = \frac{Q - U_c \cdot S_c \cdot \left(T_{c_{anterior}} - T_{h_{anterior}}\right)}{M_c} \quad [2.48]$$

$$\frac{\partial T_h}{\partial t} = \frac{T_{h_{actual}} - T_{h_{anterior}}}{t_s} = \frac{U_c \cdot S_c \cdot \left(T_{c_{anterior}} - T_{h_{anterior}}\right) - U_p \cdot S_p \cdot \left(T_{h_{anterior}} - T_e\right)}{M_h} \quad [2.49]$$

Se despeja $T_{c_{actual}}$ y $T_{h_{actual}}$:

$$T_{c_{actual}} = T_{c_{anterior}} + \frac{Q - U_c \cdot S_c \cdot \left(T_{c_{anterior}} - T_{h_{anterior}}\right)}{M_c} \, t_s \quad [2.50]$$

$$T_{h_{actual}} = T_{h_{anterior}} + \frac{U_c \cdot S_c \cdot \left(T_{c_{anterior}} - T_{h_{anterior}}\right) - U_p \cdot S_p \cdot \left(T_{h_{anterior}} - T_e\right)}{M_h} \, t_s \quad [2.51]$$

Donde t_s es el tiempo de muestreo, $T_{c_{anterior}}$ es la temperatura anterior a la que se encontraba la resistencia calorífica, $T_{c_{actual}}$ es la temperatura actual del calefactor, $T_{h_{anterior}}$ es la temperatura anterior a la que se encontraba el interior de la caja y $T_{h_{actual}}$ es la temperatura actual de la caja. La ecuación [2.50] y [2.51] permite actualizar la temperatura de la resistencia calorífica en función del tiempo (dT_c/dt) y la temperatura del interior de la caja en función del tiempo (dT_h/dt).

Cada vez que se actualizan los valores de las ecuaciones [2.50] y [2.51], la temperatura del sensor T_m (ver ecuación [2.30]) cambia de valor. La potencia calorífica Q (ver ecuación [2.35]) cambia de valor cada vez que se actualizan las ecuaciones [2.50] y [2.51], pero para la potencia calorífica hay que tener en cuenta las ecuaciones del PID (ver ecuaciones [2.31], [2.32], [2.33] y [2.34]).

3.2 INTERFAZ GRÁFICA

En este apartado veremos cómo funciona la interfaz gráfica de usuario que se ha creado, ¿qué función desempeña cada uno de los botones? ¿qué resultados se pueden obtener? ¿Qué avisos pueden aparecer?, etc.

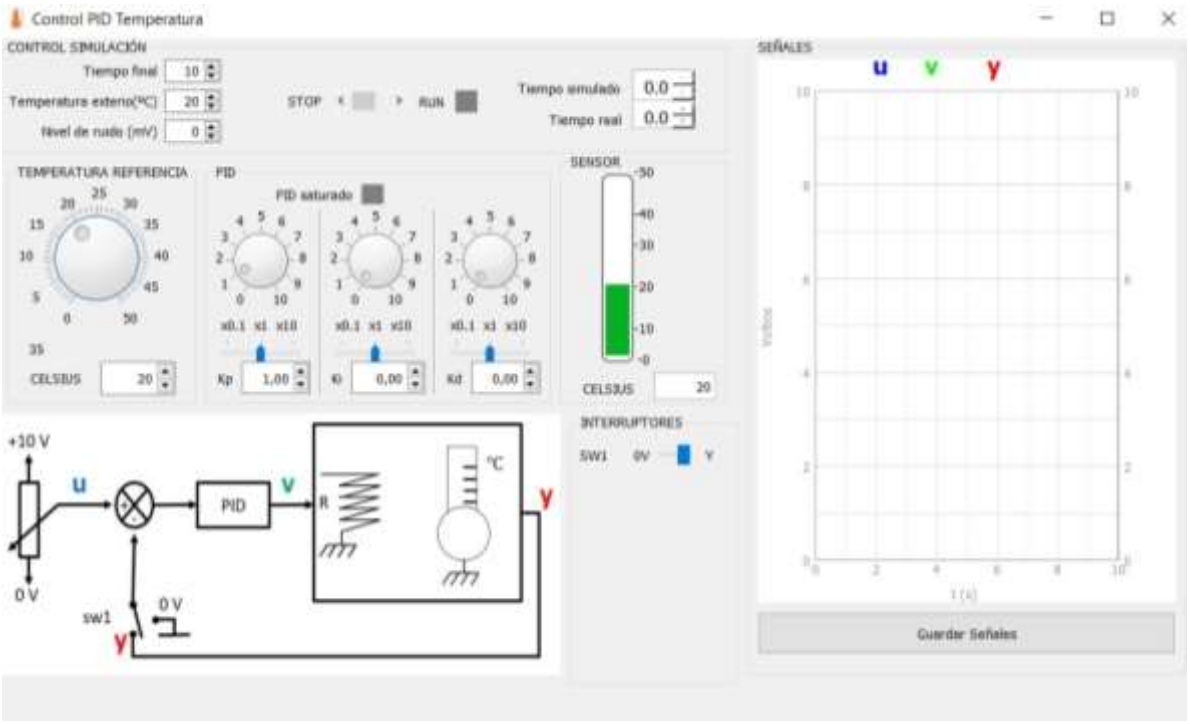


Figura 3.1. Pantalla principal de ejecución del programa.

En la **Figura 3.1** se puede ver la apariencia que tiene la interfaz de usuario una vez que se ejecuta la aplicación. En esta pantalla aparecen una serie de botones, con los que se podrán hacer una serie de ajustes antes de empezar a simular el proceso.

Inicialmente todos los botones están activos para que se pueda cambiar su valor. Desglosando la Figura 3.1 se procede a hablar sobre la zona "CONTROL DE SIMULACIÓN" (ver **Figura 3.2**)

Figura 3.2. Control de simulación antes de simular.

En la zona de "CONTROL DE SIMULACIÓN" aparecen los siguientes botones:

- Tiempo final: es un SpinBox que indica el tiempo de simulación, se expresa en segundos. Tiene un valor inicial predeterminado de 10 segundos que se podrá cambiar.
- Temperatura exterio(℃): es un SpinBox que indica la temperatura a la que se encuentra el exterior de la caja. Actúa como perturbación en el sistema.
- Nivel de ruido (mV): es un DoubleSpinBox que indica la cantidad de ruido que se tiene a la entrada del sistema, esto provocará ciertas anomalías a la hora de representar las señales.
- STOP/RUN: es un ScrollBar que al hacer clic encima de él se activa la simulación. Al iniciar la simulación el cuadradito gris oscuro que se encuentra al lado del label "RUN" cambiará de color y se pondrá de color verde, indicando que la simulación ha iniciado.
- Tiempo simulado: es un contador ascendente que indica cuanto tiempo de simulación lleva en segundos.
- Tiempo real: es un contador ascendente que indica el tiempo de simulación en segundos y en tiempo real.

Botones que se encuentran en la zona "TEMPERATURA REFERENCIA" (ver **Figura 3.3**):

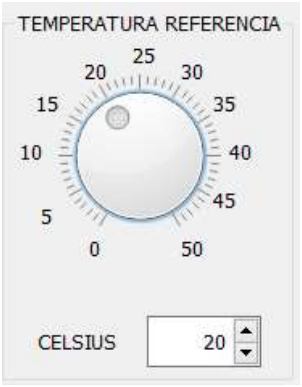


Figura 3.3. Temperatura de referencia.

En esta se encuentran dos objetos con los que se puede interactuar y están sincronizados, ya que al cambiar el valor de uno de ellos por defecto también se cambiará el valor del otro. Un Dial (rueda) que puede cambiar su valor girando. Las mismas características tiene el SpinBox la diferencia reside en que para cambiar su valor se debe escribir por teclado o utilizar las flechas ascendentes y descendentes que tiene al lado del número.

Figura 3.4. Constantes del controlador PID (kp, ki, kd).

Botones que se encuentran en la zona "PID" (ver **Figura 3.4**). En esta zona se encuentra el controlador PID, es una zona simétrica porque existen tres Diales, tres Sliders y tres DoubleSpinBox:

- Dial: toma valores de 0 a 10 por defecto. Los tres diales estarán a cero salvo el dial que corresponde a la parte proporcional del controlador PID que tomará el valor de 1.
- Slider: sirve como multiplicador. Toma valores de 0.1, 1 y 10. Inicialmente todos los multiplicadores tomarán el valor de 1. Este multiplicador sirve para hacer más grande los valores de las constantes del controlador PID, permitiendo que tomen un valor máximo de 100 o por el contrario haciéndoles más pequeños hasta tomar un valor de 0.01.
- DoubleSpinBox: esta sincronizado con el Dial, el valor que tome el Dial aparecerá reflejado en el DoubleSpinBox y viceversa.
- PID saturado: es un cuadradito que inicialmente estará en gris, pero según comience la simulación si el PID alcanza los valores de 0 o 10 V se pondrá de color rojo (indicando que el PID está saturado) y si está dentro del rango estará de color verde.

El primer Dial (donde aparece "kp") sirve para dar valores a la constante proporcional del controlador proporcional (P). El Dial que se encuentra en medio (donde aparece "ki") tomará el valor de la constante de integración del controlador integral (I) y, por último, el Dial (donde aparece "kd") es el que se encarga de dar valores a la constante derivativa del controlador derivativo (D).

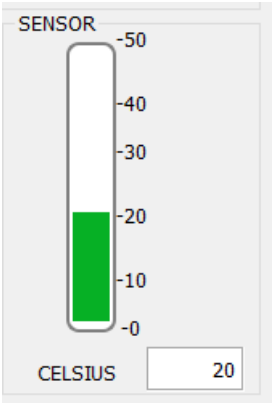


Figura 3.5. Sensor de temperatura antes de iniciar la simulación.

La zona "Sensor" (ver **Figura 3.5**). En esta zona se muestra el valor de salida del sistema que varía en función del tiempo. Este resultado cambia dependiendo de los valores que se le den en las zonas que se han visto anteriormente. Componentes que aparecen en esta zona:

- ProgressBar: tiene forma de termómetro e indica como cambia la temperatura en función del tiempo en el interior de la caja. Los resultados que arroja están expresados en grados Celsius.
- TextEdit: tiene forma de cuadrado y en él se reflejan los grados Celsius del interior de la caja.

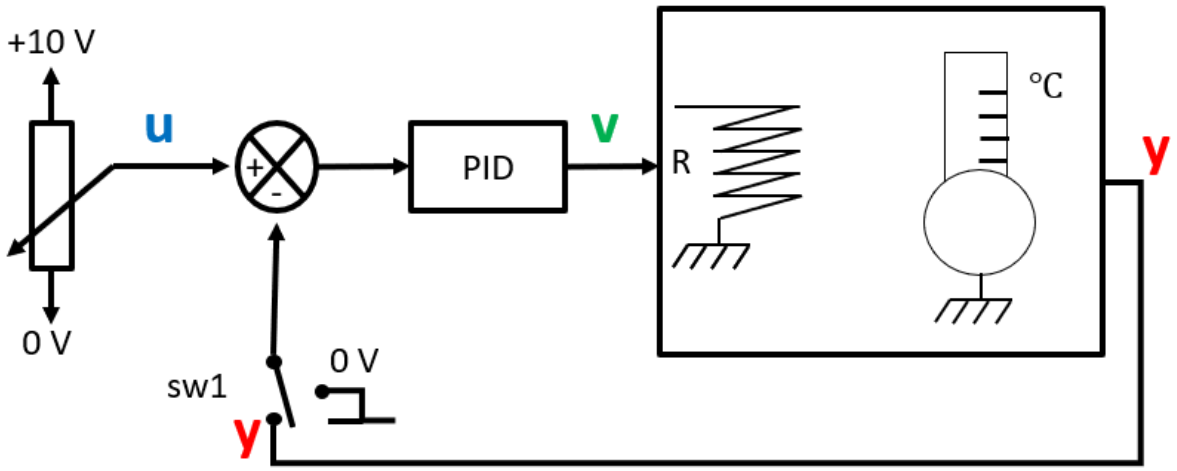


Figura 3.6. Diagrama de bloques del sistema.

En esta parte (ver **Figura 3.6**) se utiliza un Label para poder utilizar una fotografía en la que se representa todo el ciclo del sistema, donde se puede ver la entrada del controlador PID (u, en color azul), la salida del controlador PID (v, en color verde) y la salida del proceso (y, en color rojo).

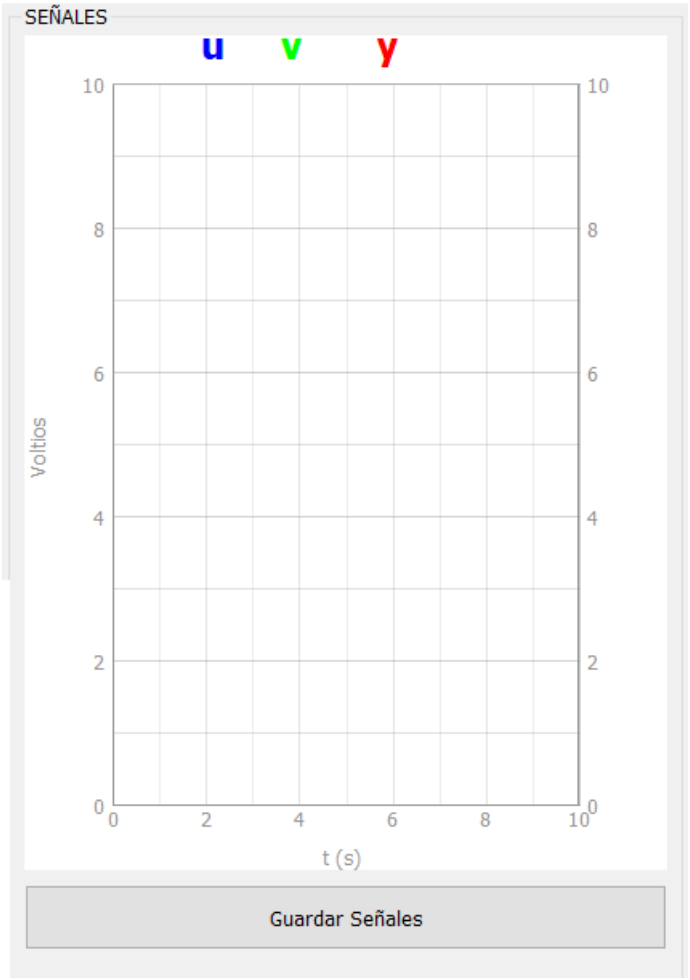


Figura 3.7. Gráfico en el que se representan las señales (u, v, y) del diagrama de bloques antes de iniciar la simulación.

En esta última zona "SEÑALES" (ver **Figura 3.7**) mediante el uso de un PlotWidget se representa las señales que se han mencionado anteriormente en el diagrama de bloques (ver **Figura 3.6**) que cambiarán en función del tiempo o según los cambios que se hagan durante la simulación (perturbaciones). En esta última zona también se encontrará un PushBotton con el nombre "Guardar Señales" que permitirá guardar las señales que se muestrean para poder trabajar con ellas en otros programas como Matlab.

3.3 CÓDIGO

Para que la GUI pueda utilizarse y que funcione correctamente, se utiliza código Python. Deben mirarse cursos de cómo utilizar las librerías "pyqt5", "numpy" y "pyqtgraph", además de entender cómo funciona la aplicación "Spyder" y "Qt Designer".

Las primeras líneas de código en Python suelen ser las de importar librerías. En el código se han importado librerías como “sys”, “pyqt5”, “pyqtgraph”, “numpy” y “time”

- Sys: es el módulo encargado de proveer variables y funcionalidades que están directamente relacionadas con el intérprete.
- Pyqt5: relaciona el código Python con el Qt Designer.
- Pyqtgraph: sirve para hacer gráficos en tiempo real a velocidades muy altas.
- Numpy: da soporte para crear vectores y matrices multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.
- Time: proporciona un conjunto de funciones para trabajar con fechas y horas.

En el código Python se utilizará dos funciones muy importantes, “tic” y “sat”:

- Tic: se utiliza para contar el tiempo de simulación.
- Sat: se utiliza para capar los valores que exceden cierto voltaje, temperatura o cualquier tipo de variable.

Una clase muy importante en el código Python es la del “PID”, la cual arrojará el valor que adquiere la tensión (en voltios) para alimentar a la resistencia calorífica. Esta tensión aumentará o disminuirá su valor en función del error.

Otra clase importante es la del “proceso” en la cual se modifican los valores de temperatura de la resistencia calorífica, la temperatura del interior de la caja y la temperatura del sensor en función del tiempo.

En la clase “Ventana” están todas las funciones que se activan en cuanto el usuario interacciona con alguno de los botones de la GUI, es decir, dependiente del botón que el usuario accione, se activará una función u otra porque en la clase “ventana” se encarga de llamar a esas funciones.

Al final de la clase “Ventana” se encuentra la función “button_guardar_clicked” en la cual se guardan las señales una vez que se han simulado para que se puedan utilizar en otros programas. Esta función se activará cuando el usuario accione el botón “Guardar Señales” de la GUI una vez finalizada la simulación.

Antes y durante la simulación el usuario podrá interactuar con una serie de botones, en cuanto el usuario active uno de los botones, en el código la clase “Ventana” hará un llamamiento a la función que corresponda, por ejemplo, si el usuario mueve el dial de referencia para marcar una temperatura en el código se activará la función “dial_referencia_movido”, esta función se encarga de almacenar el

valor que arroja el dial para que se puedan hacer los cálculos durante la simulación. Dependiendo de los botones con los que el usuario interactúe se activará una función u otra.

La función más importante que puede activar el usuario es la del “scrollBar_start_movido” la cual activará la simulación y realizará todos los cálculos que sean necesarios para poder simular las señales en tiempo real.

3.4 APLICACIÓN

Una vez visto las ecuaciones que se necesitan para simular el funcionamiento de los componentes del sistema y finalizado el código Python que sirve para poner en marcha la aplicación, se utiliza la librería “pyinstaller” de Python, para convertir un archivo (.py) a un archivo (.exe). Esto permitirá utilizar la aplicación en cualquier ordenador que tenga sistema operativo Windows. De esta manera la aplicación será portable y no habrá necesidad de tener que instalar Python para poder utilizarse.

Figura 3.8. Carpeta que contiene el archivo ejecutable.

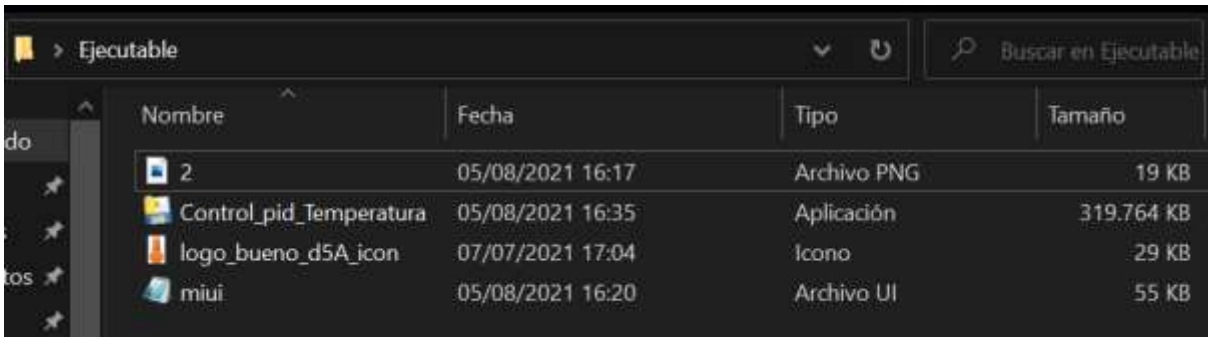


Figura 3.9. Contenido de la carpeta ejecutable.

Figura 3.10. Logotipo de la aplicación.

Como se puede apreciar en la **Figura 3.9** en la carpeta ejecutable, se encuentran 4 archivos. El primer archivo “2” es un (PNG), una imagen, que contiene el diagrama de bloques (ver **Figura 3.6**). El archivo “miui.ui” es la interfaz de usuario que se crea con el Qt Designer (ver **Figura 3.11**). El archivo “logo_bueno_d5A_icon” contiene la imagen del logotipo de la aplicación (ver Figura 3.10) y por último, el archivo ejecutable “Control_pid_Temperatura” que contiene todo el código para que la aplicación pueda funcionar. Si cualquiera de estos archivos no se encuentra en la misma carpeta la aplicación no podrá funcionar. Al intentar abrir el ejecutable que se encuentra en la carpeta ejecutable se nos abrirá la aplicación (ver **Figura 3.1**).

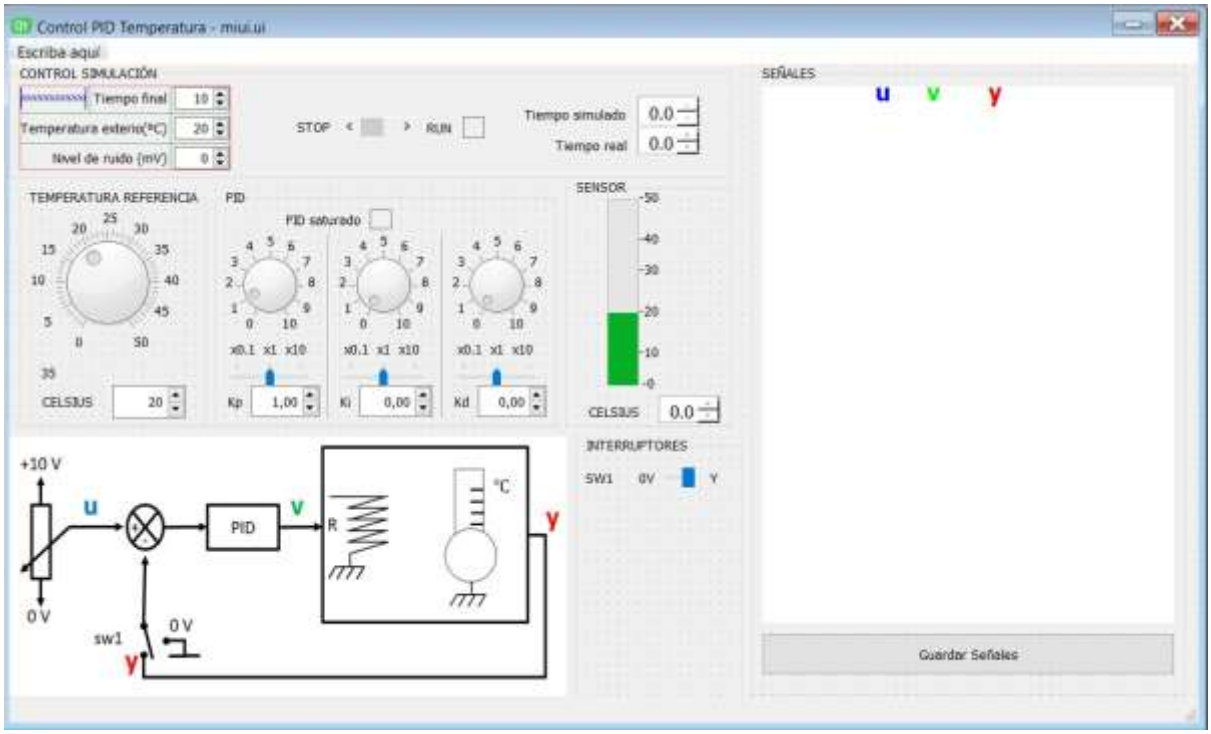


Figura 3.11. Interfaz de usuario creada con Qt Designer.

4 TRABAJOS FUTUROS

En este capítulo se hablará de las posibles mejoras o de ciertas variantes que se pueden hacer en el proyecto para que tenga un mejor funcionamiento y sea más completo, pero que por falta de tiempo no se ha implementar.

4.1 IMPLEMENTAR UNA REFRIGERACIÓN

Uno de los principales inconvenientes de la planta de control se da a la hora de tener que enfriar la temperatura en el interior de la caja ya que el enfriamiento se hace de forma natural y tarda bastante. Para poder hacer el enfriamiento del sistema existen dos opciones: enfriamiento con un ventilador o enfriamiento con una célula Peltier.

Si se implementase el modelo matemático de un ventilador se obtendría sus ecuaciones y se podrían utilizar en el código. Al conseguir las ecuaciones y con el uso del controlador PID se podría bajar la temperatura del interior de la caja de una manera más rápida y controlada.

Para poder implementar las ecuaciones matemáticas de un ventilador hay que tener en cuenta que un ventilador tiene un motor que al activarse hace girar unas aspas para que pueda sacar o meter aire en el interior de un recinto. Al sacar aire se estaría trabajando con caudales, se tendría que relacionar la salida del caudal del aire caliente con la entrada de caudal de aire frío y la subida o bajada de temperatura según el reemplazo de nuevo aire en el interior de la caja. Como mínimo se necesitan dos ecuaciones más: una que simule el arranque de un motor y otra que relacione los caudales de entrada y salida de aire con el aumento o disminución de la temperatura.

Tratando de solucionar este problema de enfriamiento se encontró las especificaciones de un ventilador que tiene una gran potencia y que puede entrar en la caja que se ha diseñado (ver **Figura 4.1**). Las especificaciones más importantes para escoger un ventilador son:

- Tamaño.
- Caudal de aire (CFM).
- Presión estática (mmH2O).
- Velocidad del ventilador (RPM).
- Voltaje, Intensidad, consumo.

Figura 4.1. Ventilador Delta QFR1212GHE.[25]

Ficha técnica del ventilador Delta: [25]

Una alternativa al ventilador es la utilización de una célula Peltier. Con la utilización de una célula Peltier se facilitan los cálculos porque no se tendría que usar caudales para enfriar el interior de la caja, sino que se utilizaría la disminución de temperatura en el interior de la caja.

Con el implemento del modelo matemático de una célula Peltier se podría sacar sus ecuaciones y utilizarlas en el código. Al conseguir las ecuaciones y con el uso del controlador PID se podría bajar la temperatura del interior de la caja.

Para poder implementar las ecuaciones matemáticas de una célula Peltier hay que tener en cuenta que una célula Peltier tiene dos caras, una de ellas calienta y la otra enfría. Lo que interesa es situar la cara de la célula que enfría hacia el interior de la caja y la parte que calienta tiene que estar apuntando al exterior de la caja. El interior de la caja, al recibir una temperatura inferior a la que el sensor arroja, provocaría que en las ecuaciones de variación de temperatura se añada una nueva variable que provocará una disminución de la temperatura. En este caso como mínimo se necesitará una ecuación que simule la variación de temperatura de la célula Peltier y una modificación en las demás ecuaciones con esta nueva variable.

4.2 IMPLEMENTACIÓN FÍSICA

Este proyecto inicialmente está planteado como si se fuera a construir una planta en forma física, se mencionan todos los materiales que se necesitan con sus propiedades y características reales, por lo que en un futuro este proyecto se podría utilizar para crear una planta de control de temperatura real.

4.3 CREAR UN SISTEMA DISCRETO

El sistema resuelto está en continuo porque las ecuaciones que se utilizan para el controlador PID están en continuo. Si fuéramos capaces de eliminar los diales del PID de la interfaz de usuario y cambiarlo por un TextEdit en el que el usuario pueda escribir las ecuaciones en discreto que quiere utilizar, el abanico de posibilidades para que el alumno pueda practicar sería mucho más amplio.

4.4 CREAR OTRAS PLANTAS DE CONTROL

Como se ha visto a lo largo del proyecto sí que se puede crear una planta de simulación de procesos que trate de replicar una planta real, por lo que se podría crear otro tipo de plantas, en vez de plantas de control de temperatura se podría crear una planta de control de llenado de un tanque de agua, una planta de control de equilibrio en una posición, una planta de control de apertura y cierre de una válvula, etc. Cuantas más plantas de simulación de procesos en formato virtual y portable se puedan crear menos será la necesidad de tener que bajar al laboratorio porque se podrá hacer desde la comodidad de nuestras casas.

5 CONCLUSIONES

En este proyecto se ha realizado la implementación de una planta de procesos térmica en formato virtual para que sea portable y que se pueda utilizar en cualquier lugar sin la necesidad de tener que utilizar internet o instalar programas adicionales.

Se ha conseguido obtener las ecuaciones necesarias que simulan el funcionamiento de los componentes que se han utilizado (sensor, resistencia, etc.)

Mediante la utilización de QT Designer se ha conseguido diseñar una interfaz de usuario intuitiva para que el alumno la pueda utilizar a su gusto y mediante la implementación de un código Python se ha relacionado la interfaz gráfica con las ecuaciones para que todo esto tenga un funcionamiento similar al de una planta de procesos física y con una simulación en tiempo real.

Por último, para que el alumno pueda entender cómo funciona la aplicación, dispone de un manual de usuario y de un libro de prácticas para que pueda asentar los conocimientos que ha adquirido en clase.

Uno de los inconvenientes del sistema es su lento enfriamiento porque a la hora de calentar el interior de la caja se hace de manera efectiva y rápida, pero a la hora de enfriar el interior de la caja, como el enfriamiento se hace de forma natural, tarda mucho. La solución sería poner un dispositivo de enfriamiento (ventilador, célula Peltier, etc.) que se encarguen de enfriar el aire caliente del interior de la caja. Al poner un dispositivo de enfriamiento se podría hacer un control del sistema óptimo.

6 BIBLIOGRAFÍA

[1] Edibon, “Aplicaciones y sistemas industriales,” 2020. https://www.edibon.com/es/planta-de-control-de-procesos-industrial-controlada-desde-computador-pc#descripciongeneral.

[2] L. L. Constantine and L. A. D. Lockwood, “Software for use: a practical guide to the models and methods of usage-centered design,” *SIGCHI Bulletin*, vol. 32, no. 1. Addison Wesley, pp. 111–114, 1999, Accessed: Jul. 08, 2021. [Online]. Available: http://portal.acm.org/citation.cfm?id=301248.

[3] Tiobe, “index | TIOBE - The Software Quality Company,” *Tiobe*, 2020. https://www.tiobe.com/tiobe-index/ (accessed Jun. 29, 2021).

[4] MathWorks, “Comparación entre MATLAB y Python: principales razones para elegir MATLAB - MATLAB & Simulink,” *MathWorks*, 2019. https://es.mathworks.com/products/matlab/matlab-vs-python.html (accessed Jul. 08, 2021).

[5] “Python vs. Matlab for Electrical Engineers | EEWeb Community.” https://www.eeweb.com/python-vs-matlab-for-electrical-engineers/ (accessed Jul. 08, 2021).

[6] (The Pyzo Team), “Python Vs MATLAB,” 2020. https://www.programaenlinea.net/python-vs-matlab/ (accessed Jul. 08, 2021).

[7] N. Community, “NumPy Reference Manual,” 2019. https://numpy.org/doc/stable/reference/ (accessed Jul. 09, 2021).

[8] M. Community, “matplotlib.pyplot — Matplotlib 3.4.2 documentation,” 2019. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html (accessed Jul. 09, 2021).

[9] Q. Group, “LA EMPRESA,” in *Wirtschaftsspanisch*, 2014.

[10] Qt, “Qt Designer Manual,” *Qt Documentation*, 2021. https://doc.qt.io/qt-5/qtdesigner-manual.html (accessed Jul. 09, 2021).

[11] A. Kuchling, “LJ Interviews Guido van Rossum | Linux Journal,” *1998*. https://www.linuxjournal.com/article/2959 (accessed Jul. 09, 2021).

[12] Python Software Foundation, “Welcome to Python.org,” 2016. https://www.python.org/

(accessed Jul. 09, 2021).

[13] B. Kuo, *Sistemas de Control Automático,7ed,931p.pdf*, Séptima ed. New York, 1996.

[14] G. F. Franklin, *Feedback Control of Dynamic Systems*, Abbas emam. California, 1994.

[15] K. Ogata, *Ingeniería de Control Moderna*, Tercera ed. Minneapolis, 1970.

[16] J. L. R., “POTENCIOMETRO - Qué es, como funciona y aplicaciones.” https://como-funciona.co/un-potenciometro/ (accessed Jul. 23, 2021).

[17] TTElectronics, “Rotary Potentiometer P160 Series,” pp. 1–5, 2019, Accessed: Jul. 23, 2021. [Online]. Available: www.ttelectronics.com/bi-technologies.

[18] Adajusa, “Controlador de temperatura termostato DTA4848R0 DELTA OMRON | ADAJUSA | precio,” 2019. https://adajusa.es/reles-de-control-y-proteccion-termostatos-sondas-de-temperatura/controlador-de-temperatura-digital-48x48.html?gclid=Cj0KCQjwlMaGBhD3ARIsAPvWd6gtpF2uREeNnSrBtkjrz3yj7d0T4b8RLyVKhO3IAUFcyzSIY4DmuowaAiz1EALw_wcB (accessed Jul. 26, 2021).

[19] P. Details, “Datasheet RS Pro Test Lead Wire Single Core 36m RS Stock No : 714-1729 Product Details Specifications :”

[20] Texas Instruments, “LM35 Precision Centigrade Temperature Sensors 1FEATURES DESCRIPTION,” p. 38, 2017, [Online]. Available: www.ti.com.

[21] TOPE, “Alambre de nicrom_Cr20Ni80,” 2017. https://www.topeintl.com/alambre-de-nicrom.html (accessed Jul. 28, 2021).

[22] “Alambre de Nicromo.” http://alambre-nicrom.blogspot.com/ (accessed Jun. 29, 2021).

[23] P. J. Rapin, J. Jacquard, P. Jacquard, and P. J. Rapin, *Instalaciones frigoríficas*. Marcombo, 1997.

[24] “Conductividad térmica - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Conductividad_térmica (accessed Jun. 29, 2021).

[25] G. P. Battery, “Specification for Approval 客户承认书,” no. 0, pp. 1–14, 2012.

7 ANEJO DE PROGRAMACIÓN

En este capítulo, se muestra todo el código del programa implementado.

7.1 IMPORTAR LIBRERÍAS

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication, QFileDialog
from pyqtgraph import QtGui
from PyQt5 import uic
import pyqtgraph as pg
import numpy as np
from time import time, sleep
import scipy.io
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.2 INICIALIZACIÓN Y DEFINICIÓN DE LAS VARIABLES

```
#valor que indica la tensión máxima:
VMAX = 10.0
#Para contar el tiempo:
def tic():
    return time()

def toc(ti):
    return time()-ti
#_____________________________________________________________________#
#_____________________________________________________________________#
#Funcion de saturación:
def sat(x, xmin, xmax):
    #si la x es menor que la xmin, nos devuelve el valor de xmin:
    if x < xmin:
        return xmin
    #si la x es mayor que la xmax, nos devuelve el valor de xmax:
    elif x > xmax:
        return xmax
    #si cumple con las condiciones nos quedamos con la x:
    else:
        return x
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.3 CLASE DEL CONTROLADOR PID

```
#_____________________________________________________________________#
#_____________________________________________________________________#
###Ecuaciones del PID:
class PID():
```

```
    def __init__(self):
        #Inicializamos el error, el error integrativo y el voltaje
del PID:
        self.e = 0.0
        self.ie = 0.0
        #ESto es la salida del PID:
        self.v = 0.0
    #Creamos una función que se actualiza:
        #ts:tiempo de muestreo
        #u:es la entrada que nos dan por el spin.Box
        #y:es la salida actual
        #kp:constante proporcional que nos la da el spin.Box
        #ki:constante integral que nos la da el spin.Box
        #kd:constante deribativa que nos la da el spin.Box
    def update(self, ts, u, y, kp, ki, kd):
        #error=Entrada - Salida:
        e = u - y
        #Corregimos el error:
        self.v = kp*e
        #Si nos dan un valor para la constante ki:
        if ki > 0.0:
            #sumamos e igualamos el error integrativo a:
            self.ie += (self.e+e)*0.5*ts
            self.v += ki*self.ie
        #Si no tenemos ki:
        else:
            self.ie = 0.0
        #Si nos dan la kd:
        if kd > 0.0:
            #error actual - el error anterior / tiempo actual; de:
es el error derivativo
            de = (e-self.e)/ts
            self.e = e
            self.v += kd*de
        #Llamamos a la función de saturación: def sat(x, xmin,
xmax):
            #x:valor del voltaje actual
            #xmin=-VMAX=-10 voltios
            #xmax=VMAX=10 voltios
        self.v = sat(self.v, 0.0, VMAX)
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.4 CLASE DEL PROCESO

```
#_____________________________________________________________________#
#_____________________________________________________________________#

class PROCESO():
    def __init__(self, T):
        self.Th = T
```

```
        self.Tc = T
        self.Tm = T #Temperatura inicial del sensor
        #Cuanto más pequeña sea la resistencia menos tardara el
sistema en llegar a la temperatura deseada:
        #self.R = 5.0 #ohmnios
        self.R = 2.3 #ohmnios
        self.Uc_Sc = 0.3834 #cal/s*Celsius
        self.Mc = 0.4264 #cal/Celsius
        self.Up_Sp = 4.93*10**-3 #cal/s*Celsius
        self.Te = T #celsius
        self.Mh = 1.0449 #cal/Celsius
        #salida actual del sistema, inicialmente es cero:
        self.v = 0.0

    def update(self, ts, vpid, Te):
        # print('Te =', Te)
        # print('self.Te =', self.Te)
        V = sat(vpid, 0.0, VMAX)
        #Ecuaciones del sistema
        q = 0.24*(V**2/self.R)#Tercera ecuación (calorías/s)
        Tc_incremento = (q- self.Uc_Sc*(self.Tc-
self.Th))/(self.Mc)#Cuarta ecuación (Temperatura incrementada)
        Th_incremento = ((self.Uc_Sc*(self.Tc-self.Th))-
self.Up_Sp*(self.Th-self.Te))/(self.Mh)
        #Actualizamos variables:
        self.Th += Th_incremento*ts
        self.Tc += Tc_incremento*ts
        #actualizamos la variable de salida del sistema:
        #Hay que capar al sensor para que no muestre más de 10
voltios, es decir 50 grados máximo:
        self.Tm =sat( 0.95*self.Th + 0.05*self.Tc, 0.0, 50.0)
#Primera ecuación (Celsius)
        #Pasamos la salida de grados Celsius a voltios, con la
ecuación de la recta:
        #self.v = sat((self.Tm-20.0)*0.5, 0.0, VMAX)
        self.v = sat((self.Tm-0.0)*0.2, 0.0, VMAX)
        #vref = (u-0.0)*0.2
        # print('Tm=', self.Tm)
        # print('V=', self.v)
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.5 CLASE CONTROL DE WIDGETS

```
#_____________________________________________________________________#
#_____________________________________________________________________#
#Clase heredada de QMAinWindow (Constructor de ventanas)
class Ventana(QMainWindow):
    def __init__(self):
        #Iniciar el objeto QMainWindow
        QMainWindow.__init__(self)
```

```
        #Cargar la configuración del archivo ui en el objeto
        uic.loadUi("miui.ui", self)
        #Las señales que activan la gráfica:
        self.widget_senales.setBackground('w')
        self.widget_senales.setLabel('left', 'Voltios')
        self.widget_senales.setLabel('right', ' ')
        self.widget_senales.setLabel('bottom', 't (s)')
        self.widget_senales.setTitle(" ")
        self.widget_senales.showGrid(x = True, y = True)
        self.widget_senales.setXRange(0.0, 10.0, padding = 0)
        self.widget_senales.setYRange(0.0, 10.0, padding = 0)
        #Inicializamos la luz del PID y la del Encendido:
        self.label_luzpid.setStyleSheet('background-color: rgb(128,
128, 128);')
        self.label_luzon.setStyleSheet('background-color: rgb(128,
128, 128);')
        #ESta es la mejor de momento:
        self.progressBar.setStyleSheet('QProgressBar {border: 2px
solid grey;border-radius:8px;padding:1px}"QProgressBar::chunk
{background:yellow}")')
        #ESte es el más parecido a un termómetro:
        # self.progressBar.setStyleSheet('QProgressBar {border: 2px
solid grey;border-radius:8px;padding:1px} QProgressBar::chunk
{background:red}')
        al = self.textEdit_tiemposimulado.alignment()
        self.textEdit_salida.setText(str(round(20,1)))
        self.textEdit_salida.setAlignment(al)
        #Para centrar la aplicación en el centro de la pantalla:
        self.centerOnScreen() #(OPCIONAL)
        #Si el dial de referencia se mueve vamos a la función
dial_referencia_movido:

self.dial_referencia.valueChanged.connect(self.dial_referencia_movid
o)
        #Si el spinBox de referencia cambia de valor vamos a la
función spinBox_referencia_cambiado:

self.spinBox_referencia.valueChanged.connect(self.spinBox_referencia
_cambiado)
        #Si el dial_kp cambia de valor vamos a la función
dial_kp_movido:
        self.dial_kp.valueChanged.connect(self.dial_kp_movido)
        #Si el dial_ki cambia de valor vamos a la función
dial_ki_movido:
        self.dial_ki.valueChanged.connect(self.dial_ki_movido)
        #Si el dial_ki cambia de valor vamos a la función
dial_kd_movido:
        self.dial_kd.valueChanged.connect(self.dial_kd_movido)
        self.slider_kp.valueChanged.connect(self.slider_kp_movido)
        self.slider_ki.valueChanged.connect(self.slider_ki_movido)
        self.slider_kd.valueChanged.connect(self.slider_kd_movido)
```

```
        #Si el spinBox_kp cambia de valor vamos a la función
spinBox_kp_cambiado:

self.spinBox_kp.valueChanged.connect(self.spinBox_kp_cambiado)
        #Si el spinBox_ki cambia de valor vamos a la función
spinBox_ki_cambiado:

self.spinBox_ki.valueChanged.connect(self.spinBox_ki_cambiado)
        #Si el spinBox_kd cambia de valor vamos a la función
spinBox_kd_cambiado:

self.spinBox_kd.valueChanged.connect(self.spinBox_kd_cambiado)
        #Si movemos el scrollBar para activar el sistema, vamos a la
función scrollBar_start_movido:

self.scrollBar_start.valueChanged.connect(self.scrollBar_start_movid
o)
        #Cuando se pulse el botón de guardar gráfica:

self.pushButton_guardarsenales.clicked.connect(self.button_guardar_c
licked)
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.6 FUNCIÓN GUARDAR SEÑALES

```
#_____________________________________________________________________#
#_____________________________________________________________________#
    #Para guardar el grafico:
    def button_guardar_clicked(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getSaveFileName(self,"Guardar
señales","","Matlab Files (*.mat);;All Files (*)", options=options)
        filename = fileName.split('.')[0]
        if len(filename) > 0:
            scipy.io.savemat(filename+'.mat', dict(t = self.t, u =
self.vrefn, v = self.vpidn, y = self.voutn))
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.7 FUNCIÓNES DE LOS WIDGETS

```
#_____________________________________________________________________#
#_____________________________________________________________________#
#Clase heredada de QMAinWindow (Constructor de ventanas)
    #Función que centra la aplicación en mitad del escritorio:
(OPCIONAL)
    def centerOnScreen (self):#(OPCIONAL)
        resolution = QtGui.QDesktopWidget().screenGeometry()
```

```
        self.move(int((resolution.width()-
self.frameSize().width())/2),
                  int((resolution.height()-
self.frameSize().height())/2))
#_____________________________________________________________________#
#_____________________________________________________________________#
    #Funcion que hace que el spinBox obtenga el valor del dial de
referencia:
    def dial_referencia_movido(self):

self.spinBox_referencia.setValue(self.dial_referencia.value())
    #Funcion que sincroniza el spinBox con el dial de referencia:
    def spinBox_referencia_cambiado(self):
        self.dial_referencia.blockSignals(True)

self.dial_referencia.setValue(self.spinBox_referencia.value())
        self.dial_referencia.blockSignals(False)

#_____________________________________________________________________#
#_____________________________________________________________________#
    #Funcion que hace que el spinBox obtenga el valor del dial_kp:

    def dial_kp_movido(self):
        #hay que multiplicar por 0.1 el valor del dial ya que el
dial arroja valores de números enteros
        dial = self.dial_kp.value()*0.1
        slider = self.slider_kp.value()
        kp = dial*10**slider
        #El spinbox con decimales coge el valor de la kp
        self.spinBox_kp.blockSignals(True)
        self.spinBox_kp.setValue(kp)
        self.spinBox_kp.blockSignals(False)

    def slider_kp_movido(self):
        dial = self.dial_kp.value()*0.1
        slider = self.slider_kp.value()
        kp = dial*10**slider
        self.spinBox_kp.blockSignals(True)
        self.spinBox_kp.setValue(kp)
        self.spinBox_kp.blockSignals(False)

    #Funcion que hace que el spinBox obtenga el valor del dial_ki:
    def dial_ki_movido(self):
        dial = self.dial_ki.value()*0.1
        slider = self.slider_ki.value()
        ki = dial*10**slider
        self.spinBox_ki.blockSignals(True)
        self.spinBox_ki.setValue(ki)
        self.spinBox_ki.blockSignals(False)

    def slider_ki_movido(self):
        dial = self.dial_ki.value()*0.1
```

```
        slider = self.slider_ki.value()
        ki = dial*10**slider
        self.spinBox_ki.blockSignals(True)
        self.spinBox_ki.setValue(ki)
        self.spinBox_ki.blockSignals(False)

    #Funcion que hace que el spinBox obtenga el valor del dial_kd:
    def dial_kd_movido(self):
        dial = self.dial_kd.value()*0.1
        slider = self.slider_kd.value()
        kd = dial*10**slider
        self.spinBox_kd.blockSignals(True)
        self.spinBox_kd.setValue(kd)
        self.spinBox_kd.blockSignals(False)

    def slider_kd_movido(self):
        dial = self.dial_kd.value()*0.1
        slider = self.slider_kd.value()
        kd = dial*10**slider
        self.spinBox_kd.blockSignals(True)
        self.spinBox_kd.setValue(kd)
        self.spinBox_kd.blockSignals(False)

#_____________________________________________________________________#
#_____________________________________________________________________#
    #Funcion en la que el spinBox_kp hace cambiar el slider_kp y al
dial_kp:
    def spinBox_kp_cambiado(self):
        self.slider_kp.blockSignals(True)
        self.dial_kp.blockSignals(True)
        kp = self.spinBox_kp.value()
        if kp < 1:
            exp = -1
        elif kp < 10:
            exp = 0
        else:
            exp = 1
        self.slider_kp.setValue(exp)
        self.dial_kp.setValue(int(10*kp/10**exp))
        self.slider_kp.blockSignals(False)
        self.dial_kp.blockSignals(False)
    #Funcion en la que el spinBox_ki hace cambiar el slider_ki y al
dial_ki:
    def spinBox_ki_cambiado(self):
        self.slider_ki.blockSignals(True)
        self.dial_ki.blockSignals(True)
        ki = self.spinBox_ki.value()
        if ki < 1:
            exp = -1
        elif ki < 10:
            exp = 0
        else:
```

```
            exp = 1
        self.slider_ki.setValue(exp)
        self.dial_ki.setValue(int(10*ki/10**exp))
        self.slider_ki.blockSignals(False)
        self.dial_ki.blockSignals(False)
    #Funcion en la que el spinBox_kd hace cambiar el slider_kd y al
dial_kd:
    def spinBox_kd_cambiado(self):
        self.slider_kd.blockSignals(True)
        self.dial_kd.blockSignals(True)
        kd = self.spinBox_kd.value()
        if kd < 1:
            exp = -1
        elif kd < 10:
            exp = 0
        else:
            exp = 1
        self.slider_kd.setValue(exp)
        self.dial_kd.setValue(int(10*kd/10**exp))
        self.slider_kd.blockSignals(False)
        self.dial_kd.blockSignals(False)
#_____________________________________________________________________#
#_____________________________________________________________________#
```

7.8 INICIO DE LA SIMULACIÓN

```
#_____________________________________________________________________#
#_____________________________________________________________________#

    def scrollBar_start_movido(self):
        if self.scrollBar_start.value() == 1:
            #Deshabilito los spinBox del tiempo final y ruido,
cuando accione el inicio del sistema y no se pueda cambiar el valor:
            self.spinBox_tiempofinal.setEnabled(False)
            self.spinBox_ruido.setEnabled(False)
            #Ponemos a verde la luz start y limpiamos la gráfica:
            self.label_luzon.setStyleSheet('background-color: rgb(0,
255, 0);')
            self.widget_senales.clear()
            #cogemos el color azul y con un grosor de 2 para dibujar
la señal Vref:
            penvref = pg.mkPen(color = (0, 0, 255), width = 2)
            #cogemos el color verde y con un grosor de 2 para
dibujar la señal Vpid:
            penvpid = pg.mkPen(color = (0, 255, 0), width = 2)
            #cogemos el color rojo y con un grosor de 2 para dibujar
la señal Vout:
            penvout = pg.mkPen(color = (255, 0, 0), width = 2)
            #alignment() Devuelve la alineación actual del párrafo:
            al = self.textEdit_tiemposimulado.alignment()
```

```
            #Guardamos el tiempo final que nos dan por pantalla en
la variable tfin:
            tfin = float(self.spinBox_tiempofinal.value())
            #creamos un objeto de tipo clase para dar un valor
inicial, final, un número de muestras y el tipo de dato de la
matriz:
            self.t = np.linspace(0.0, tfin, int(tfin*1000), dtype =
'float')
            #creamos un objeto de tipo clase para nos devuelva una
matriz de ceros:
            self.vrefn = np.zeros_like(self.t)
            self.voutn = np.zeros_like(self.t)
            self.vpidn = np.zeros_like(self.t)
            #Lo dejamos preparado para poder dibujar los valores de
x, y, con el nombre de u y con el bolígrafo de penvref o penvpid o
penvout:
            hvref = self.widget_senales.plot([], [], name = "u", pen
= penvref)
            hvpid = self.widget_senales.plot([], [], name = "v", pen
= penvpid)
            hvout = self.widget_senales.plot([], [], name = "y", pen
= penvout)
            #Hacemos visible el rango de X con el
padding(relleno)=0:
            self.widget_senales.setXRange(0.0, tfin, padding = 0)
            #Inicializar variables:
            #Llamamos a la función PID:
            pid = PID()
            #Llamamos a la función proceso:
            proceso = PROCESO(self.spinBox_texterior.value())
            tt = 0.0
            k = 0
            #inicializamos el tiempo:
            t0 = tic()
            #sleep() aceptará un tiempo con una fracción no nula
            sleep(0.01)
            #mientras la variable tt sea menor a tfin permanecemos
en el bucle:
            while tt < tfin:
                #si ponemos en off la barra de encendido, salimos
del bucle:
                if self.scrollBar_start.value() == 0:
                    break
                #tiempo anterior será tt:
                t_anterior = tt
                #El tiempo tt se dará en la función toc():
                tt  = toc(t0)
                #tiempo resultante: ESTA ES UNA VARIABLE IMPORTANTE
                ts = tt - t_anterior
                #aumentamos +1 la variable k:
                k += 1
                #Añadimos ruido:
```

```
                nl = self.spinBox_ruido.value()*0.001
                #los valores de las constantes del PID, que se
actualizan:
                kp = self.spinBox_kp.value()
                ki = self.spinBox_ki.value()
                kd = self.spinBox_kd.value()
                sw4 = float(self.slider_sw4.value())
                #Grados de inclinación:
                #Lo llamaremos u porque es nuestra entrada, la
temperatura que queremos:
                u = self.spinBox_referencia.value() #trabajamos con
grados Celsius
                proceso.Te = float(self.spinBox_texterior.value())
                #actualizamos la clase pid: def update(self, ts, u,
y, kp, ki, kd):
                    #ts:tiempo actual
                    #u:es la entrada que nos dan por el spin.Box
                    #y:es la salida actual
                    #kp:constante proporcional que nos la da el
spin.Box
                    #ki:constante integral que nos la da el spin.Box
                    #kd:constante derivativa que nos la da el
spin.Box
                #vref = (u-20.0)*0.5
                vref = (u-0.0)*0.2

                pid.update(ts, vref, sw4*proceso.v, kp, ki, kd)
                    #proceso.update(self, ts, vpid, u):
                        #ts:tiempo actual
                        #pid.v:el valor de voltaje que devuelve el
PID=vpid
                        #u: temperatura de la entrada, la que
queremos
                proceso.update(ts, pid.v, proceso.Te)
                #para que avance el tiempo:
                self.t[k] = tt
                #a los grados le metemos ruido para simular que no
es exacto y nos devuelve la tensión de referencia:
                self.vrefn[k] = sat(vref + np.random.randn()*nl,
0.0, VMAX)
                    #self.vrefn[k] = vref
                #al voltaje del pid le añadimos el ruido:
                self.vpidn[k] = sat(pid.v + np.random.randn()*nl, -
VMAX, VMAX)
                    #self.vpidn[k] = pid.v
                #al voltaje de salida del proceso le añadimos ruido:
                self.voutn[k] = sat(proceso.v +
np.random.randn()*nl, 0.0, VMAX)
                    #self.voutn[k] = proceso.v
                #si el voltaje esta fuera de rango, la luz del pid
se pone en rojo, indicando que está saturado y si está dentro se
mantiene en verde:
```

```
                if pid.v == 0.0 or pid.v == 10.0:
                    self.label_luzpid.setStyleSheet('background-
color: rgb(255, 0, 0);')
                else:
                    self.label_luzpid.setStyleSheet('background-
color: rgb(0, 255, 0);')
                #Multiplicamos por 10 para pasar los decimales a
entero y que el progresbar no avance a saltos:
                self.progressBar.setValue(int(proceso.Tm*1000))
#celsius
                #mostrar el valor de la temperatura del sensor:

self.textEdit_salida.setText(str(round(proceso.Tm,1)))
                self.textEdit_salida.setAlignment(al)
                #mostramos en el spin.box tipo texto la cuenta
atras:
                self.textEdit_tiemposimulado.setText(str(round(tt,
1)))
                #alineamos el texto en el centro:
                self.textEdit_tiemposimulado.setAlignment(al)
                #mostramos en el spin.box tipo texto la cuenta
atras:
                self.textEdit_tiemporeal.setText(str(round(toc(t0),
1)))
                #alineamos el texto en el centro:
                self.textEdit_tiemporeal.setAlignment(al)

                #Guardamos los valores de entrada:
                hvref.setData(self.t[:k], self.vrefn[:k])
                #Guardamos valores de la salida del pid:
                hvpid.setData(self.t[:k], self.vpidn[:k])
                #Guardamos la salida del sistema:
                hvout.setData(self.t[:k], self.voutn[:k])
                QApplication.processEvents()
            #Actualizamos los valores:
            self.t = self.t[:k]
            self.vrefn = self.vrefn[:k]
            self.vpidn = self.vpidn[:k]
            self.voutn = self.voutn[:k]
            #Ponemos en gris las luces led:
            self.label_luzpid.setStyleSheet('background-color:
rgb(128, 128, 128);')
            self.label_luzon.setStyleSheet('background-color:
rgb(128, 128, 128);')
            #self.label_Calentador.setStyleSheet('background-color:
rgb(128, 128, 128);')
            #desabilidamos el valor del scrollBar para que se pueda
desactivar:
            self.scrollBar_start.setValue(0)
            self.progressBar.setValue(0)
            #Habilito los spinBox:
            self.spinBox_tiempofinal.setEnabled(True)
```

```
            self.spinBox_ruido.setEnabled(True)
            #self.spinBox_zonamuerta.setEnabled(True)
            #muestro por pantalla el tiempo simulado y el número de
muestras:
            #print('tiempo real = ', toc(t0))
            #print('numero de muestras = ', len(self.t))
#_____________________________________________________________________#
#_____________________________________________________________________#
#Instancia para iniciar una aplicación
app=QApplication(sys.argv)
#crear un objeto de la clase
_ventana=Ventana()
#mostrar la ventana
_ventana.show()
#Ejecutar la aplicación, tienes 2 formas:
sys.exit(app.exec_()) #app.exec_()
```